

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

DIAGNOSTIC DES RÉSEAUX D'INTERCONNEXIONS PROGRAMMABLES
DANS LES CIRCUITS INTÉGRÉS À L'ÉCHELLE DE LA TRANCHE DE
SILICIUM

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE

PAR

SION GONTRAN

NOVEMBRE 2016

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

Avertissement

La diffusion de ce mémoire se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.07-2011). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche, le professeur Yves Blaquière, pour m'avoir tout d'abord accepté dans son équipe de recherche, puis pour ses conseils avisés et son soutien tout au long de cette maîtrise. De même, je tiens à remercier mon codirecteur de recherche, le professeur Yvon Savaria, pour son écoute et ses conseils.

Je remercie aussi tous mes coéquipiers du laboratoire GRM de l'École Polytechnique de Montréal pour leur bonne humeur et leur partage. Plus spécialement Sylvain Charasse et Safa Berrima qui ont contribué à la mise en place de l'environnement de test.

Je tiens à remercier tous les employés de l'Université du Québec à Montréal, de même que de l'École Polytechnique de Montréal qui m'ont apporté leur aide.

Je ne saurai assez remercier ma moitié, Astrik, pour son soutien et sa patience dans cet apprentissage.

Finalement, je tiens à remercier ma mère et mes sœurs, pour leurs soutiens sans faille qui m'ont permis de finir ce travail.

RÉSUMÉ

Ce mémoire présente un algorithme de diagnostic d'un réseau d'interconnexions programmable (*Field Programmable Interconnect Network*, FPIN) dans un circuit intégré à l'échelle de la tranche de silicium (*Wafer Scale Integrated Circuit*, WSIC). Ce WSIC est au cœur du projet de recherche DreamWaferTM regroupant plusieurs universités canadiennes, de même que des partenaires industriels. Ce projet vise à élaborer une plateforme de prototypage rapide pour les systèmes électroniques. Ce dispositif est comparable à un circuit imprimé reprogrammable et intelligent, permettant d'interconnecter les composants électroniques déposés à sa surface conformément aux spécifications fournies.

La surface de silicium de ce FPIN de 200 mm de diamètre implique la présence inévitable de pannes et demande donc l'utilisation d'un algorithme de diagnostic permettant de détecter et de localiser ces pannes afin d'appliquer une stratégie de tolérance aux pannes. Le temps de diagnostic de ce FPIN a un coût temporel non négligeable et croissant avec l'augmentation de la couverture des pannes. Ainsi l'algorithme proposé a pour but de minimiser le temps de test et diagnostic, tout en maximisant la couverture des pannes. Le test et la validation d'un prototype du WSIC ont été réalisés et ont permis de tester l'algorithme de diagnostic du FPIN proposé.

Ce mémoire présente l'élaboration de l'environnement de test, de même que de l'outil logiciel développé pour y appliquer les algorithmes de diagnostic proposés. L'environnement de test a aussi permis de valider la fonction de construction de chaînes JTAG reconfigurables et tolérantes aux pannes. L'algorithme de diagnostic proposé utilise une approche de configuration en diagonale afin de réduire le cône

d'influence des tests des liens d'interconnexions du FPIN, réduisant le nombre de configurations nécessaires de $O(n^4)$ à $O(n^3)$ par rapport aux méthodes existantes, où n est le nombre de liens. De plus, l'algorithme proposé augmente la couverture des pannes en ajoutant les courts-circuits au diagnostic des commutateurs programmables (*crossbar*) par rapport aux algorithmes existants. L'algorithme proposé a été testé sur un réseau de 32×32 cellules du prototype, et les tests, dont l'efficacité est calculée en nombre de cycles de la machine à états finis du contrôleur JTAG, ont montré une réduction du temps de diagnostic de 112 fois par rapport à ceux des algorithmes existants.

ABSTRACT

This thesis presents test and diagnosis algorithms of a defect tolerant field programmable interconnect network (*FPIN*) in a wafer scale integrated circuit (WSIC). The WSIC is the core of the DreamWaferTM research project involving several Canadian Universities and industrial partners. The project aims to develop a platform for rapid electronic system prototyping. This system is comparable to a reprogrammable and smart printed circuit board, allowing to interconnect electronic components deposited on the surface in accordance with the specifications.

Due to the *FPIN* silicon area of 200 mm diameter, defects are inevitable. The use of test and diagnosis algorithms to detect and locate faults is a must to apply fault-tolerance strategy. Time of diagnosis implies a sizeable temporal cost that grows with the fault coverage. Hence, the proposed test and diagnosis algorithms try to reduce test and diagnostic time, while keeping the maximal fault coverage. Prototype testing and validation of the WSIC were performed and allowed to test the proposed diagnosis algorithm.

This thesis shows the elaboration of the test environment, as well as the software tool developed to apply the proposed algorithms. The test environment also allowed to validate the reconfigurable and defect tolerant JTAG chain. The proposed diagnosis algorithm of the *FPIN*'s links uses a diagonal configuration to reduce the cone of influence, allowing test's parallelization. This algorithm reduced the number of configurations from $O(n^4)$ to $O(n^3)$ compared to existing methods, where n is the number of links. The proposed diagnosis algorithm of the *FPIN*'s crossbars add short defect to the fault coverage compared to existing methods. The proposed diagnosis

algorithms was applied on a reticule prototype of 32×32 cells, and results show a 112-fold test time reduction, where times are calculated in terms of the number of cycles of the JTAG finite state machine.

TABLE DES MATIÈRES

| | |
|--|-----|
| RÉSUMÉ | V |
| ABSTRACT | VII |
| LISTE DES FIGURES | XI |
| LISTE DES TABLEAUX | XIV |
| LISTE DES SIGLES ET ABRÉVIATIONS | XV |
| INTRODUCTION | 1 |
| CHAPITRE I | |
| REVUE DE LITTÉRATURE | 5 |
| 1.1 Évolution des systèmes intégrés | 5 |
| 1.2 Terminologie du test et diagnostic | 10 |
| 1.3 Test et diagnostic des circuits imprimés | 12 |
| 1.3.1 Norme IEEE-Std 1149.1 | 12 |
| 1.3.2 Modèles de pannes | 15 |
| 1.3.3 Techniques de génération de vecteur de test | 18 |
| 1.4 Test et diagnostic des circuits logiques programmables | 21 |
| 1.5 Résumé | 25 |
| CHAPITRE II | |
| REVUE DE LA PLATEFORME DE PROTOTYPAGE RAPIDE | 27 |
| 2.1 Plateforme de prototypage rapide de systèmes électroniques | 28 |
| 2.2 Suite logicielle de la plateforme de prototypage rapide | 30 |
| 2.3 Interposeur intelligent à l'échelle de la tranche de silicium | 32 |
| 2.4 Système de configuration tolérant aux pannes | 35 |
| 2.5 Réseau d'interconnexions à l'échelle de la tranche de silicium | 40 |
| 2.6 Résumé | 45 |

| | |
|---|-------------------------------------|
| CHAPITRE III | |
| TEST ET DIAGNOSTIC DU RÉSEAU D'INTERCONNEXIONS | |
| PROGRAMMABLE..... | 47 |
| 3.1 Algorithme de diagnostic des commutateurs dans le réseau d'interconnexions | |
| programmable..... | 48 |
| 3.2 Algorithme de diagnostic des liens du FPIN..... | 54 |
| 3.3 Analyse du temps de test et de la complexité de l'algorithme de diagnostic | 61 |
| 3.3.1 Estimation du temps de diagnostic des commutateurs..... | 62 |
| 3.3.2 Estimation du temps de diagnostic des liens du FPIN..... | 63 |
| 3.3.3 Analyse de complexité..... | 65 |
| 3.4 Résumé..... | 67 |
| CHAPITRE IV | |
| RÉSULTATS..... | 69 |
| 4.1 Description de l'environnement de test..... | 70 |
| 4.1.1 Connexions d'alimentation et de communication au MiniWaferIC | 70 |
| 4.1.2 Test de fonctionnement du MiniWaferIC..... | 72 |
| 4.2 Outil logiciel pour l'analyse et l'automatisation..... | 77 |
| 4.2.1 Présentation générale de l'outil logiciel..... | 77 |
| 4.2.2 Fonction de remise à zéro..... | 78 |
| 4.2.3 Fonction de construction de la chaîne JTAG..... | 79 |
| 4.2.4 Fonction de configuration des autres registres..... | 81 |
| 4.3 Résultats et analyses..... | 81 |
| 4.3.1 Construction de chaînes JTAG..... | 82 |
| 4.3.2 Test de l'algorithme de diagnostic de <i>crossbars</i> | 84 |
| 4.3.3 Test de l'algorithme de diagnostic de liens..... | Error! Bookmark not defined. |
| 4.4 Conclusion..... | 88 |
| CHAPITRE V | |
| CONCLUSION..... | 91 |
| BIBLIOGRAPHIE..... | 97 |
| APPENDICE A | |
| ARTICLE DE CONFÉRENCE..... | 105 |

LISTE DES FIGURES

| Figure | Page |
|---|------|
| Figure 1.1 Système sur tranche (Poupon <i>et al.</i> , 2009) | 8 |
| Figure 1.2 Réseau sur puce (Yu et Ampadu, 2012) | 8 |
| Figure 1.3 Structure de circuit logique programmable (Yue et Dongfang, 2002) | 9 |
| Figure 1.4 Origine de défaillance | 12 |
| Figure 1.5 Architecture de balayage aux frontières (Jarwala et Chi, 1989) | 14 |
| Figure 1.6 Topologie de réseau | 19 |
| Figure 1.7 Système mixte de test interne/externe (Yue et Dongfang, 2002) | 22 |
| Figure 1.8 Structure de conception en vue du test (DFT) dans les circuits logiques programmables (Doumar et Ito, 2003; Wang <i>et al.</i> , 2006) | 23 |
| Figure 1.9 Configuration des commutateurs en vue du test des interconnexions (Doumar et Ito, 2003) | 24 |
| Figure 2.1 Modèle éclaté de la plateforme de prototypage (André <i>et al.</i> , 2011) | 29 |
| Figure 2.2 Plateforme de prototypage rapide de systèmes électronique WaferBoard™ | 30 |
| Figure 2.3 Flot de conception de la suite logicielle | 31 |
| Figure 2.4 Architecture du réseau d'interconnexions programmable (Norman <i>et al.</i> , 2008) | 33 |
| Figure 2.5 Schéma d'une cellule du WaferIC™ | 34 |
| Figure 2.6 Module WaferIC/PowerBlock, appelé MiniWaferIC | 34 |
| Figure 2.7 Diagramme d'états du contrôleur TAP (www.xjtag.com/about-jtag/jtag-a-technical-overview/) | 37 |
| Figure 2.8 Construction d'une chaîne JTAG dans le FPIN | 39 |
| Figure 2.9 Construction d'une chaîne JTAG évitant les défauts | 41 |

| | | |
|-------------|--|----|
| Figure 2.10 | Interconnexions d'une cellule dans le FPIN | 43 |
| Figure 2.11 | Schéma d'une cellule du WaferIC™ | 43 |
| Figure 3.1 | Cellule du VLAIC et ses chaînes de registres..... | 48 |
| Figure 3.2 | Algorithme de diagnostic d'un <i>crossbar</i> à N entrées..... | 50 |
| Figure 3.3 | Configurations et vecteurs de test pour un <i>crossbar</i> à N=3 entrées | 51 |
| Figure 3.4 | Modèle de pannes d'un <i>crossbar</i> 3×3 | 52 |
| Figure 3.5 | Cône d'influence des deux algorithmes..... | 55 |
| Figure 3.6 | Algorithme précédemment proposé pour le test des liens (Basile-Bellavance, 2009) | 55 |
| Figure 3.7 | Algorithme de diagnostic en diagonale des liens du FPIN sur un réseau de 7×7 cellules | 57 |
| Figure 3.8 | Algorithme de diagnostic diagonalisé des liens du FPIN | 59 |
| Figure 3.9 | Test des configurations complémentaires..... | 60 |
| Figure 4.1 | Module WaferIC/PowerBlock, appelé MiniWaferIC | 71 |
| Figure 4.2 | Module MiniWaferIC (a) et son environnement de test (b)..... | 71 |
| Figure 4.3 | Alimentations de l'environnement de test | 71 |
| Figure 4.4 | Schématique de l'environnement de test du MiniWaferIC..... | 72 |
| Figure 4.5 | Connexions du PowerBlock..... | 73 |
| Figure 4.6 | Environnement de test du MiniWaferIC..... | 73 |
| Figure 4.7 | Système de refroidissement du MiniWaferIC..... | 76 |
| Figure 4.8 | Flot générique de l'outil logiciel pour le diagnostic du FPIN..... | 78 |
| Figure 4.9 | Exemple de chaîne configurée par la commande [1 1 2 2 3 3 4 3]..... | 80 |
| Figure 4.10 | Vecteurs de configuration des registres Dest pour la chaîne JTAG [1 2 3 3]..... | 80 |
| Figure 4.11 | Signaux JTAG générés pour la construction de la route [1 2 3 3] | 83 |
| Figure 4.12 | Confirmation de la construction d'une chaîne JTAG complète par l'extraction de la signature..... | 84 |

Figure 4.13 Signaux JTAG de l'application d'un test sur 4 cellules du diagnostic des commutateurs du FPIN 86

Figure 4.14 Visualisation des test de l'algorithme de diagnostic des liens du FPIN 86

LISTE DES TABLEAUX

| Tableau | Page |
|---|------|
| Tableau 1.1 Séquence de vecteurs du test « <i>counting sequence algorithm</i> » pour un FPIN de 8 connexions indépendantes..... | 19 |
| Tableau 1.2 Séquence de vecteurs du test « <i>Modified counting sequence algorithm</i> » pour un FPIN de 8 connexions indépendantes..... | 19 |
| Tableau 2.1 Liste des commandes du registre d'instruction | 42 |
| Tableau 3.1 Syndrome produit par des pannes simples diagnosticables pour un <i>crossbar</i> 3×3 | 53 |
| Tableau 3.2 Résumé des temps de test et de la complexités des l'algorithmes de diagnostic pour un réseau de 32×32 cellules | 66 |
| Tableau 4.1 Commandes d'intérêt des modes de registres..... | 81 |
| Tableau 4.2 Liste des contributions..... | 89 |

LISTE DES SIGLES ET ABRÉVIATIONS

| | |
|------|---|
| BGA | Ball Grid Array |
| CMP | Chip MultiProcessor |
| DFT | Design for test |
| FPGA | Field programmable gate array |
| FPIN | Field Programmable Interconnection Network |
| FSM | Finite State Machine |
| IC | Integrated Circuit |
| IoC | Interconnexion on Chip |
| IP | Intellectual Property |
| JTAG | Joint Test Action Group (norme IEEE 1149.1) |
| MCM | MultiChip Module |
| NoC | Network on Chip |
| NRST | Neutral Test Reset |
| PCB | Printed Circuit Board |

| | |
|-------|------------------------------------|
| RTCK | Returned Test Clock |
| SiP | System in Package |
| SMD | Surface Mounted Device |
| SMD | Surface mounted device |
| SoC | System on Chip |
| SoP | System on Package |
| SoW | System on Wafer |
| TAP | Test Acces Port |
| TCK | Test Clock |
| TDI | Test Data Input |
| TDO | Test Data Output |
| TRST | Test Reset |
| TTM | time to market |
| VLAIC | Very Large Area Integrated Circuit |
| VLSI | Very large scale integration |
| WSIC | Wafer Scale Integrated Circuit |

INTRODUCTION

L'évolution des systèmes électroniques, à la fois dictée par des besoins de performance et de réduction des coûts, tend vers une intégration toujours plus forte, et avec elle, à une augmentation des contraintes. Les cartes de circuits imprimés (*Printed Circuit Board*, PCB) actuels utilisent des composants de plus en plus intégrés avec l'utilisation de boîtier à matrice de billes (*Ball Grid Array*, BGA) avoisinant les 2000 billes de connexions pour des boîtiers de 45 mm² dans le cas des Virtex 7 de Xilinx par exemple. Ainsi, ces boîtiers peuvent demander l'utilisation de PCB pouvant dépasser les 20 couches de métallisations. Les contraintes de ce type de matériel sont :

- Un coût important proportionnel à la surface du PCB, donc au nombre de composants électroniques utilisés et à la complexité du routage qu'ils demandent.
- Une intégration et performance limitée par les contraintes physiques des matériaux utilisés.
- Un test et déverminage complexe dû aux problèmes d'accès à des signaux ou points de test pouvant être prisonniers sous des boîtiers ou entre plusieurs couches de métallisation.

Les problématiques qu'apportent ces contraintes demandent la recherche de nouveaux substrats; en particulier pour le prototypage rapide, où les coûts et le temps de conception ne peuvent être amortis par la production en série. Les tranches de silicium déjà utilisées pour les interposeurs sont apparues comme une alternative intéressante.

Alors que la surface des interposeurs n'est en général pas suffisante en comparaison à la surface demandée dans la conception des PCB, Richard Norman a proposé l'utilisation d'un circuit de la taille d'une tranche de silicium (*Wafer Scale Integrated Circuit*, WSIC) qui répond à toutes les contraintes citées plus haut. Ainsi, il introduit un nouveau concept de circuit électronique nommé WaferIC™. Le WaferIC est un type d'interposeur intelligent à l'échelle de la tranche de silicium pouvant interconnecter tous types de boîtiers déposés à sa surface, sans contraintes d'alignement, par l'utilisation des spécifications d'interconnexions (*netlist*) prédéfini par l'utilisateur. La partie intelligente de l'interposeur réside dans le réseau d'interconnexions programmable (*Field Programmable Interconnection Network*, FPIN) qui le compose, et qui permet de programmer à volonté le routage de cet interposeur de la taille d'une tranche de silicium pour permettre la réutilisation du WaferIC dans le cadre du prototypage rapide de systèmes électroniques. Cette solution a été intégrée dans un système nommé WaferBoard™, brevetée par Norman Richard et propriété de Gestion TechnoCap Inc.

Le WaferBoard est le système matériel complet de prototypage rapide de systèmes électroniques, incluant le FPIN WaferIC, de même que l'électronique nécessaire à son fonctionnement (puissance, contrôle, communication et parties mécaniques). Cette plateforme ressemble à un gaufrier, qui, ouvert, permet de déposer les composants électroniques du système à concevoir sur la surface du FPIN, et cela sans contraintes d'alignement. Puis, il suffit de fermer le couvercle pour immobiliser les composants électroniques sur le FPIN, et appliquer la pression nécessaire au contact parfait entre ces composants électroniques et la surface du FPIN.

Le WaferIC est composé de 76 réticules identiques répétés sur toute la surface de la tranche de silicium. Chaque réticule est composé d'une mer de 32×32 cellules identiques et interconnectées entres-elles formant un réseau d'interconnexions à l'échelle de la tranche de silicium. Ces cellules sont elles-mêmes composées de

16 Nanopads. Ces Nanopads sont des contacts permettant l'alimentation de la broche connectée à sa surface, ou le routage vers d'autre Nanopads de la tranche de silicium. Ainsi, combiné au FPIN, le routage du système peut être programmé aux souhaits de l'utilisateur pour former le système électronique désiré.

Pour permettre la configuration du FPIN, un sous-ensemble de la norme IEEE 1149.1 (*Joint Test Action Group*, JTAG) est utilisé. La différence importante de ce sous-ensemble avec la norme JTAG, est l'aspect configurable de la chaîne JTAG. Ainsi, le chemin de la chaîne JTAG entre toutes les cellules du FPIN, et d'un réticule en particulier, n'est pas défini et doit être configuré au fur et à mesure de l'utilisation et des besoins du système, au contraire de la norme JTAG qui définit des chaînes fixes. Cette fonctionnalité permet une tolérance fondamentale aux pannes dans les circuits à l'échelle de la tranche de silicium où les pannes sont inévitables dues à la surface du circuit.

La présence de pannes doit être considérée aussi bien dans le système de configuration JTAG que dans le FPIN, et un diagnostic de ce dernier doit être possible pour permettre de contourner ses pannes lors de la phase de routage. Une proposition de diagnostic du FPIN a été faite par Yan Basile-Bellavance dans (Basile-Bellavance, 2009). Dans ce mémoire, nous poursuivons ce travail.

Les objectifs de recherche de ce mémoire sont :

- augmenter la couverture des pannes du FPIN par rapport aux méthodes existantes ;
- améliorer le temps de diagnostic du FPIN par rapport aux méthodes existantes ;
- implémenter le diagnostic sur un prototype de WaferIC de 2×2 réticules connecté à un module d'alimentation et de contrôle.

Les contributions de ce mémoire sont :

- La modélisation, définition et analyse du modèle physique utilisé pour l'étude afin d'y analyser l'algorithme proposé dans ce mémoire ;
- La définition d'un algorithme réduisant le nombre de configurations nécessaire au diagnostic des liens de $O(n^4)$ à $O(n^3)$ où n est le nombre de liens ;
- La réduction du temps de test de 112 fois pour le diagnostic des pannes de court-circuit et de collé-à dans les crossbars et les liens d'un réseau de 32×32 cellules, par rapport au temps de l'algorithme existant ;
- L'augmentation de la couverture des pannes par l'ajout de la couverture des courts-circuits dans les crossbars par rapport à l'algorithme existant ;
- La publication d'un article à la conférence IOLTS 2015 (Sion *et al.*, 2015) ;
- La mise en place d'un environnement de test pour le prototype réalisé par l'équipe DreamWafer™. Cet environnement de test a été réalisé conjointement avec Sylvain Charasse ;
- La programmation d'un logiciel de génération de trames JTAG à l'intention du WaferIC ;
- La programmation d'un microprogramme pour FPGA permettant d'interfacer l'environnement de test avec le programme de contrôle.

Ce mémoire présente le travail effectué tout au long de la maîtrise. Il commence par une revue de la littérature dans le chapitre 1. Puis, en chapitre 2, une présentation approfondie du projet DreamWafer est faite. Par la suite, le chapitre 3 présente l'étude théorique de l'algorithme de diagnostic proposé. Dans le chapitre 4, une présentation des autres contributions, de même que des résultats de test est faite. Enfin, ce mémoire se termine par un résumé des contributions et des travaux effectués, de même que par une analyse critique de ceux-ci. Une proposition de travaux futurs y est faite.

CHAPITRE I

REVUE DE LITTÉRATURE

Ce mémoire présente une étude du diagnostic des réseaux d'interconnexions programmables (*Field Programmable Interconnect Network*, FPIN). L'une des utilisations majeures des FPIN est l'intégration des circuits électroniques s'appuyant sur l'utilisation d'interconnexions sur puce (*Interconnect on Chip*, IoC), et particulièrement d'interposeurs intelligents. Ainsi, la revue de littérature propose dans une première section une description de l'environnement entourant les systèmes intégrés. La seconde section définira la terminologie du test et diagnostic utilisé dans ce mémoire et correspondant à la terminologie de la littérature. Enfin, en lien avec le diagnostic des FPIN de ce mémoire, les troisième et quatrième sections présenteront l'état de l'art du test et diagnostic dans les circuits imprimés et dans les circuits logiques programmables (*Field Programmable Gate Array*, FPGA) respectivement.

1.1 Évolution des systèmes intégrés

Depuis le premier transistor des laboratoires Bell et le premier processeur 4004 d'Intel avec ses 2300 transistors, jusqu'aux processeurs d'aujourd'hui, comme les Xeon multicœurs composés de plus de 4 milliards de transistors, la complexité des systèmes électroniques croît. « Cette évolution est une conséquence de la loi de Moore (qui permet un plus grand niveau d'intégration), et de l'économie de marché où les consommateurs demandent toujours plus de fonctionnalité dans des produits plus petits, moins chers et avec une meilleure durée de vie sur batterie » (Furber et

Bainbridge, 2005). L'électronique est un compromis entre toutes les contraintes auxquelles elle fait face.

Ainsi, les contraintes physiques des cartes de circuits imprimés (*Printed Circuit Board*, PCB) ont influencé la volonté d'intégration des composants électroniques, et plus que cela, ces contraintes ont poussé l'intégration de fonctions différentes sur un même circuit intégré (*Integrated Circuit*, IC), les systèmes sur puce (*System on Chip*, SoC). Ainsi, on peut trouver aujourd'hui des systèmes intégrés comme des microcontrôleurs à base de processeur ARM par exemple, et dont le processeur est une propriété intellectuelle (*Intellectual Property*, IP) sous licence d'une compagnie et intégré dans les systèmes d'autres compagnies. Autour de ce processeur, les compagnies peuvent ainsi ajouter leurs modules comme de la mémoire, des modules de traitements du signal analogique ou numérique, des modules de communications sans fil ou filaire, des bus d'interconnexions, etc. Mais les défis principaux dans le progrès des SoC sont le design, la vérification du design, la fabrication, la propriété intellectuelle et les problèmes légaux, le temps d'arriver sur marché (*time to market*, TTM) et le prix (RAO, 2006).

Pour dépasser ces problèmes, d'autres tendances ont émergé. Au lieu d'intégrer le système sur un unique et complexe circuit intégré (SoC), la conception se fait sur de plus petits modules hautement aboutis et combinés pour reconstruire un large système en un module multi puce (*Multi-Chip Module*, MCM). C'est le « Diviser pour régner ». Cette solution bien que complexe reste avantageuse en interconnectant des dizaines de puces en une petite structure horizontale, minimisant les risques à des IC plus petits, dont le risque de panne, la complexité et le coût sont moindres.

Les systèmes comme les téléphones portables, qui demandent une plus grande intégration, ont, eux, choisis la technologie d'empilement 3-D de puces (*3-D chip stacking*) intégré dans un seul boîtier, système en boîtier (*System in Package*, SiP) (Dick, 2015). C'est la version verticale du MCM qui a l'avantage de ne pas

augmenter la surface finale de l'IC. Cependant, l'empilement 3-D augmente grandement la complexité de la mise en boîtier, car l'accès aux contacts d'entrées/sorties devient plus compliqué. On peut aussi évoquer les systèmes sur boîtier (*System on Package*, SoP) qui eux, en plus des SiP, comportent des composants passifs intégrés dans les couches minces, afin de répondre encore plus aux contraintes d'intégrations (*Modeling and optimization of SiP/SoP and packaging for electrical integrity*, 2010).

Un autre concept d'intégration hétérogène est le système sur tranche (*System on Wafer*, SoW) (Fig. 1.1), qui a pour avantage une haute résolution de lithographie grâce aux procédés du silicium, un coefficient de dilatation compatible à celui du silicium, un substrat actif et qui permet une réduction des coûts grâce à l'utilisation des technologies au niveau de la tranche de silicium (Poupon *et al.*, 2009).

Le fait de pouvoir combiner plusieurs modules dans un boîtier a comme avantages de permettre la réutilisation d'IP (« l'*IP reuse* ») et de simplifier la conception de circuit à signal mixte analogique/numérique. Ainsi, il y a une réduction du coût de conception, de la complexité des procédés et une TTM plus rapide. De plus, l'intégration sur tranche (SoW) a les avantages d'avoir une haute résolution de lithographie grâce aux procédés du silicium, et permet de s'affranchir des contacts intrapuce permettant ainsi des fréquences de fonctionnement plus importantes.

Le point central de toutes ces technologies est le réseau d'interconnexions qu'elles utilisent pour interconnecter tous les modules du circuit électronique final. Il existe deux grands types d'interconnexion, les réseaux sur puce (*Network on Chip*, NoC) et les interconnexions sur puce (*Interconnect on Chip*, IoC).

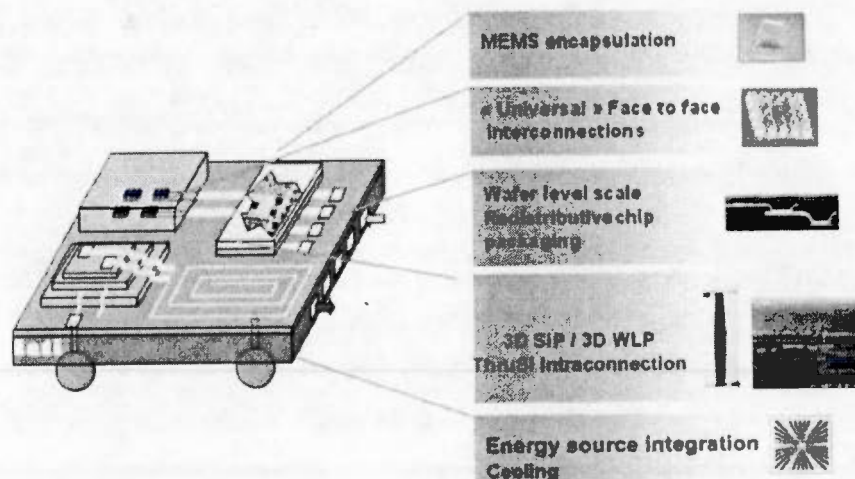


Figure 1.1 Système sur tranche (Poupon *et al.*, 2009)

Un NoC est une infrastructure de communication sur puce composée de trois composants fondamentaux (Fig. 1.2) – les interconnexions (*links*), les interfaces de réseau (*network interfaces*), et les routeurs (*routers*) (Yu et Ampadu, 2012). Ceux-ci sont majoritairement utilisés dans les SoC ou dans les puces à multiprocesseurs (*Chip MultiProcessor*, CMP) où ils permettent de gérer la complexité des interconnexions et de faciliter l'intégration de blocs IP variés. Dans ces systèmes, un routage de paquets est nécessaire pour transmettre l'information entre deux points.

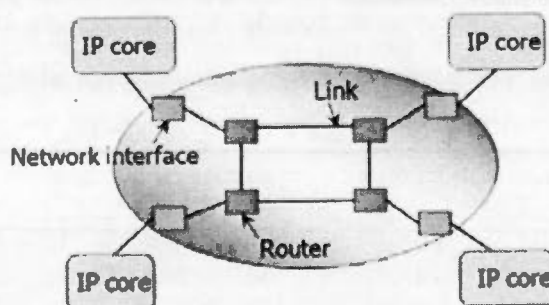


Figure 1.2 Réseau sur puce (Yu et Ampadu, 2012)

Les systèmes de type matrice de cellules de granularité différentes (Fig. 1.3), comme les FPGA, disposent d'éléments identiques sous forme de cellules et répété sur toute la surface du circuit intégré dans les directions x et y. Ces cellules sont interconnectées par un réseau d'interconnexions que l'on nomme réseau d'interconnexions programmables (*Field Programmable Interconnect Network*, FPIN). Ces systèmes de matrice de cellules définissent leurs interconnexions lors de leur configuration (c.à.d. programmation). Leur réseau d'interconnexions devient alors permanent (exception faite de la reconfiguration dynamique des FPGA) et aucun routage n'est nécessaire en fonctionnement. Comme seule la composante interconnexion est nécessaire, on parle ici de réseau d'interconnexions programmable. De même, dans les MCM, SiP et SoP, le substrat sur lequel sont déposées les puces est un réseau d'interconnexions « statique ». On parlera donc de réseau d'interconnexions au sens large (IoC).

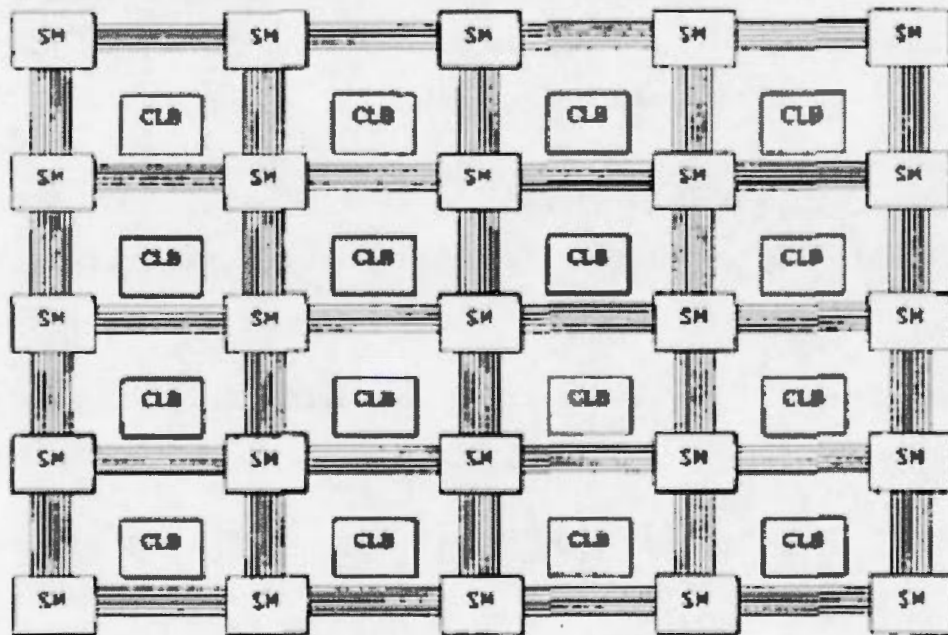


Figure 1.3 Structure de circuit logique programmable (Yue et Dongfang, 2002)

C'est dans le contexte des SoW, et plus particulièrement des circuits intégrés de très grande surface (*Very Large Area Integrated Circuit*, VLAIC) que le brevet de Richard Norman (Norman, 2008) a été développé. Ce mémoire présente ce VLAIC et le test et diagnostic de son FPIN.

La très grande surface de silicium des VLAIC engendre inévitablement la présence de pannes, et un test et diagnostic de ces circuits est une obligation. Ainsi, la section suivante présentera la terminologie du test et diagnostic.

1.2 Terminologie du test et diagnostic

Dans les systèmes de type matrice de cellules, le réseau d'interconnexions est prédominant, et représente plus de 50 %, voire plus de 80 % de la surface du circuit intégré (Doumar et Ito, 2003; McCracken et Zilic, 2002). La recherche de la fiabilité, soit la qualité d'un système de fournir le service (comportement) que l'on s'attend à recevoir de lui (Laprie, 1995), est une préoccupation première lors de la conception ainsi que lors du test et diagnostic des interconnexions. Mais pour suivre la seconde loi de Moore, dont la miniaturisation constante des composants des circuits intégrés et la réduction de la puissance sont des tendances majeures, cela induit une augmentation de la probabilité que des défauts se produisent lors de la production et une sensibilité accrue des circuits à toutes variations de procédés ou de milieux.

Laprie (Laprie, 1995) définit les attributs qui constituent la fiabilité et la robustesse d'un système par :

- La détérioration de la fiabilité: pannes (*fault*), erreurs (*error*) et défaillances (*failure*).
- L'expression de la fiabilité: l'évitement de pannes (*fault-avoidance*), la tolérance aux pannes (*fault-tolerance*), la suppression de fautes (*error-removal*) et la prévision de fautes (*error-forecasting*).

- La mesure de la fiabilité: robustesse (*reliability*), disponibilité (*availability*), maintenance (*maintainability*) et la sécurité (*safety*).

Ainsi, un court-circuit dans un circuit intégré est une panne ; la conséquence est une erreur qui reste latente (ne s'exprime pas) aussi longtemps qu'elle n'est pas activée. À son activation (utilisation du module où l'erreur se situe, et du signal ou combinaison de signaux permettant l'expression de cette erreur), l'erreur devient effective; quand cette erreur produit une ou des données erronées, ce qui affecte le service fourni, une défaillance survient (Laprie, 1995). Notons que la cause de la panne peut provenir d'un problème de conception ou spécification du système, d'une panne durant la procédure de conception, ou d'un système sans panne qui en a eu suite à des causes physiques ou environnementales (Abraham et Fuchs, 1986). En d'autres termes, une erreur est une manifestation dans le système d'une panne, et une défaillance est la manifestation d'une erreur qui affecte un service (Laprie, 1995) (Fig. 1.4). Notons qu'une panne peut être permanente, les mêmes stimuli permettront à l'erreur qui en découle, si erreur il y a, de s'activer. Mais une panne peut être intermittente, soit que l'erreur associée ne s'exprime que si des conditions particulières de température, vibration ou autre soient présentes. Une panne est caractérisée par sa nature, sa valeur, son ampleur et sa durée (Aviziens, 1976).

Pour qu'un système atteigne une fiabilité, Laprie (Laprie, 1995) dit que le système doit utiliser la combinaison d'une série de méthodes parmi l'évitement de pannes, la tolérance aux pannes, la suppression d'erreurs, et la prévision d'erreurs. L'évitement de pannes et la tolérance aux pannes peuvent être vus comme constituant de source de fiabilité (*dependability procurement*); ou comment permettre au système de fournir un service spécifique. La suppression d'erreur et la prévision d'erreur peuvent être vues comme constituant de validation de fiabilité (*dependability validation*); ou comment atteindre une confiance dans un système à ce qu'il fournisse un service spécifique (Laprie, 1995).

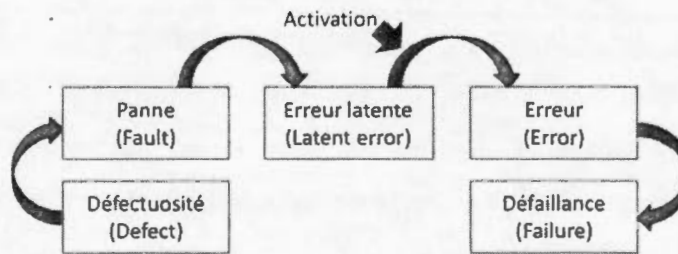


Figure 1.4 Origine de défaillance

Dans le cas du FPIN de ce mémoire, en s'appuyant sur la redondance de ses liens et de ses commutateurs programmables (*crossbar*), une stratégie de tolérance aux pannes est appliquée. Ainsi, un test et diagnostic est effectué avant la configuration du FPIN pour identifier les éléments défectueux ou fonctionnels, et cette information est utilisée par la suite pour n'utiliser que les éléments fonctionnels lors du routage. Ce mémoire présente un algorithme permettant de détecter et localiser les pannes du FPIN. Cet algorithme s'appuie sur les algorithmes utilisés pour le test et diagnostic des PCB. Ainsi, la section suivante présentera ces algorithmes.

1.3 Test et diagnostic des circuits imprimés

Les techniques de test et diagnostic des circuits imprimés ont évolué avec la complexité croissante de ces derniers. L'utilisation actuelle des circuits intégrés montés en surface (*Surface Mounted Device*, SMD), en plus du nombre croissant de couches de métallisation utilisées dans les circuits imprimés, limitent les possibilités d'accès direct à des parties des systèmes, et ainsi, peut contraindre à se passer de point de test.

1.3.1 Norme IEEE-Std 1149.1

Les PCB ont une fonction de routage, tout comme les IoC, ainsi leurs tests et diagnostics ont une très grande similitude. Aujourd'hui, avec les contraintes d'intégration, les dimensions des connexions tendent à décroître toujours plus. Ces contraintes, en plus de l'augmentation des fréquences de fonctionnement des

systemes, amènent des problèmes d'intégrité du signal. De même, les erreurs de procédé lors de la production peuvent amener à des défaillances. Ainsi le test et diagnostic des PCB, comme des IoC, est une nécessité fondamentale. Pour ce faire, la norme IEEE-Std 1149.1 ("IEEE Standard Test Access Port and Boundary Scan Architecture," 2001) a été développée. Le standard IEEE *Test Access Port and Boundary-Scan Architecture* (IEEE-Std 1149.1), est une technique de conception en vue du test (*design-for-testability*, DFT). Cette technique a été développée initialement pour le test des PCB et est connue aussi sous l'appellation abusive de JTAG pour *Joint Test Action Group*. Par la suite, cette technique a été intégrée à la conception en vue du test des circuits intégrés à très grande échelle (*Very Large Scale Integration*, VLSI), et permet aujourd'hui la configuration et l'observation de certains systèmes électronique en plus du test et diagnostic de ceux-ci.

Le concept de balayage de la frontière du circuit (*boundary-scan*) permet l'accès et le contrôle externe de toutes les entrées et sorties primaires du circuit à l'aide d'un registre à décalage formé par les cellules JTAG et d'un port d'accès de test (*Test Access Port*, TAP) (Fig. 1.5). Le TAP, en plus de contrôler le protocole JTAG, facilite le standard de communication de test entre différents circuits du système. Dans ce standard, le TAP utilise une entrée pour le registre à décalage, le TDI (*Test Data In*), une sortie de ce registre, le TDO (*Test Data Out*), et deux signaux de contrôle de la machine à états finis (*Finite-State Machine*, FSM) du contrôleur JTAG, l'horloge TCK (*Test Clock*) et le contrôle de test TMS (*Test Mode Select*). Optionnellement, on retrouve la broche TRST (*Test Reset*) pour la remise à zéro du TAP. Ces signaux permettent d'accéder à 2 types de données :

- Les registres d'instruction du TAP, qui permettent de définir le comportement du JTAG.
- Les registres de données, qui permettent de configurer ou extraire des valeurs du système.

Aujourd'hui, certains systèmes possèdent deux nouvelles broches, le retour de l'horloge pour plus de contrôle (*Returned Test Clock*, RTCK), et une remise à zéro « neutre » (*Neutral Test Reset*, NRST), soit à un état connu du programmeur.

Pour les systèmes VLSI, le concept a été adapté aussi bien pour des blocs internes des circuits que pour les broches d'entrées/sorties. De même, dans le FPIN étudié dans ce mémoire, l'utilisation de cette norme a été adaptée pour tolérer les pannes, et sera présentée dans le chapitre 2.

Cependant, la chaîne de test est composée d'une série de cellules pouvant être supérieur à 1000 et la nature sérielle de cette technologie oblige à minimiser le nombre et la longueur des tests. Effectivement, si la taille du test d'un algorithme est d'une complexité $O(\log n)$ avec n cellules, alors le temps de test dans l'environnement de chaîne de test périphérique devient $O(\log n^2)$ (Yang *et al.*, 2010).

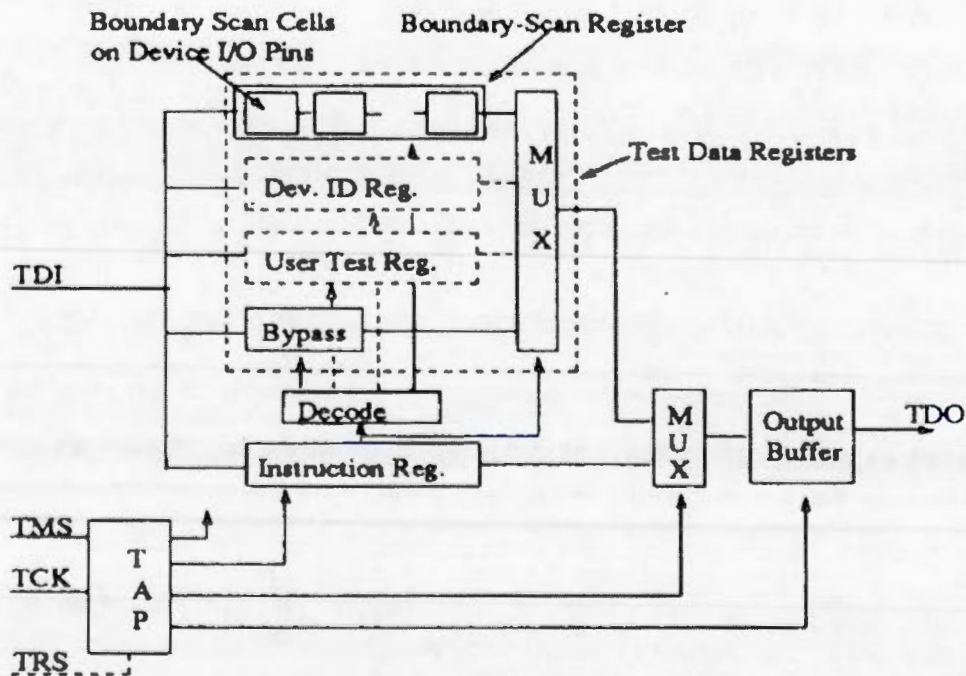


Figure 1.5 Architecture de balayage aux frontières (Jarwala et Chi, 1989)

Ainsi les algorithmes ont évolué, en jouant sur le compromis entre une couverture des pannes la plus large possible et d'un nombre de tests et de taille de vecteur le plus petit possible. Tout cela pour minimiser le temps de test et diagnostic, et donc le coût de ce test et diagnostic.

1.3.2 Modèles de pannes

Pour concevoir un système tolérant aux pannes ou pour tester et diagnostiquer le système, la base de l'étude se fonde sur une variété de modèles de pannes et d'erreurs. Un modèle de pannes décrit des défauts et des défaillances, de même que les patrons d'entrées qui les détecteront, donc appropriés pour le test. Tandis que les modèles d'erreurs décrivent les effets des défauts sur les sorties fonctionnelles et sont utiles pour la détection des erreurs en fonctionnement (*on-line*) (Abraham et Fuchs, 1986). Ainsi, un modèle de pannes semble plus approprié pour le diagnostic.

Une panne décrite au niveau des transistors, soit en expliquant les effets d'une défaut, peut décrire en profondeur les phénomènes physiques liant cause et effets entre la défaut et la panne, mais dès que l'on arrive dans des systèmes avec un millier, un million de transistors, le modèle de niveau transistor n'est plus approprié. C'est ici que la notion de niveau d'abstraction entre en jeu.

Un modèle de pannes est une description des effets d'une défaut ou d'une défaillance d'un circuit (Abraham et Fuchs, 1986). Les pannes et défaillances dans les circuits intégrés modernes peuvent être représentées comme des courts-circuits et des circuits ouverts dans les interconnexions et les circuits dégradés (Mangir, 1984). De ce fait, les modèles de pannes au niveau des transistors peuvent caractériser assez précisément les défaillances physiques (Burgess et Damper, 1984), (Gai *et al.*, 1983), (Case, 1976), (Al-Arian et Agrawal, 1987) et ("Fault Modeling," 1985).

Tandis que la classe des courts-circuits et circuits ouverts est dans la majorité des modèles de pannes, les modèles les plus précis et complexe incorporent la classe des délais. Les modèles qui incorporent la dégradation des éléments sont le plus souvent utilisés pour les circuits analogiques (Bandler et Salama, 1985).

Le modèle de pannes de circuit ouvert et court-circuit au niveau transistors modélise la plupart des défauts et défaillances physiques des circuits intégrés (Abraham et Fuchs, 1986). Tandis que Galiay (Galiay *et al.*, 1980) a montré, en utilisant un microprocesseur 4-bit, que la majorité des pannes de ce circuit ont été des courts-circuits et des circuits ouverts au niveau transistor, Courtois (Courtois, 1981) a proposé un modèle général pour les pannes de courts-circuits et les circuits ouverts au niveau transistor qui divise les pannes en trois classes, qui demande une difficulté de test progressif.

L'un des modèles le plus communément utilise est le modèle simple collé-à (*Single Stuck-At* (S@ ou SA)). Ce modèle assume que les défauts et pannes physiques peuvent être représentées comme si une ligne (entrées ou sorties) au niveau portes logiques a été bloquée à un niveau logique permanent '0' ou '1' (Abraham et Fuchs, 1986). La force de ce modèle est qu'il a été montré que les pannes au niveau transistor et système peuvent être modélisées au niveau logique (Abraham et Fuchs, 1986). Le modèle S@ est souvent référencé comme le modèle de pannes classique et offre une bonne représentation pour la majorité des types de pannes (e.g., circuit ouvert et court-circuit dans la technologie Metal Oxide Complémentaire, (*Complementary Metal Oxide Semiconductor*, CMOS)) (Lala, 2009). Certains types de pannes ont plus tendance à survenir que d'autres (Corsi, 1991; Shen *et al.*, 1985), ce qui implique que la couverture des pannes est dépendante de la probabilité de l'occurrence, comme du potentiel de la détection de ces pannes (Abraham et Fuchs, 1986). Enfin, même si toutes les pannes au niveau transistor ne peuvent pas être modélisées au niveau porte, le modèle S@ pourra quand même détecter les pannes

non modélisables, et est donc viable (Abraham et Fuchs, 1986; Banerjee et Abraham, 1984; Lala, 2009). La représentation de plusieurs pannes par le modèle multiple S@ est possible, le modèle multiple S@ assume que plusieurs lignes sont bloquées à l'état logique '1' ou '0', avec une variante, la panne unidirectionnelle, lorsque toutes les pannes sont soit S@'1', soit S@'0' (Lala, 2009).

Pour compléter le modèle S@, le modèle de court-circuit logique (*Bridging fault*) peut modéliser des pannes que ne peut pas modéliser le modèle S@. Le modèle court-circuit logique est utilisé pour décrire des pannes représentant la connexion de deux ou plusieurs lignes entre-elles au niveau portes logiques, et aboutir à deux comportements, le court-circuit logique « ET » et le court-circuit logique « OU » (Abraham et Fuchs, 1986). Ce modèle se divise en deux types, la connexion entre deux signaux entrants (*input bridging*) et la connexion entre un signal entrant et une sortie. Ce dernier peut faire osciller le circuit, ou converger vers un circuit séquentiel (Lala, 2009). Wadsack (Wadsack, 1978) a relevé un cas spécial de panne de circuit ouvert (court-circuit) au niveau transistor, qu'il a appelé panne bloquée-ouverte (*StuckOpen, StuckOn*), où le transistor peut se comporter comme une porte logique possédant une mémoire. Et une séquence particulière est nécessaire pour détecter cette panne. Cependant, (Chiang et Vranesic, 1983; Elzig, 1981) décrivent un générateur d'algorithme de tests utilisant le modèle S@ comme base pour détecter des pannes bloquées-ouvertes. Enfin, les systèmes utilisant de plus en plus de primitives, les modèles précédents sont trop détaillés pour décrire des systèmes VLSI. Alors des modèles pour les fonctions particulières du système doivent être faits pour permettre l'optimisation du test à ce niveau.

Les modèles utilisés au niveau des PCB et IoC sont des modèles niveau transistor de type court-circuit, qui peuvent être de 4 types différents ;

- court-circuit logique « ET », où l'état logique '0' est dominant ;
- court-circuit logique « OU », où l'état logique '1' est dominant ;

- court-circuit faible, où le signal flotte entre les deux états logiques ;
- court-circuit avec un signal dominant, où un signal définit l'état logique.

Ce modèle représente la panne la plus courante dans ces types de circuits. Notons que les courts-circuits logiques « ET » et « OU » sont fonction de la technologie utilisée dans les IC pour les IoC, et ne peuvent être tous deux présents si la technologie est homogène, ce qui n'est pas vrai pour les PCB qui peuvent regrouper plusieurs technologies.

Enfin, d'autres modèles sont importants :

- le collé à zéro, quand la ligne est court-circuitée à la masse ;
- le collé à un, quand la ligne est court-circuitée à l'alimentation du circuit ;
- le collé ouvert, quand le transistor est toujours ouvert ;
- le collé fermé, quand le transistor est toujours fermé ;
- le modèle de délai.

Ainsi, dans l'étude du FPIN de ce mémoire les modèles S@ et court-circuit seront analysés. Une fois le modèle de panne établi, la question des tests et de leur application se pose.

1.3.3 Techniques de génération de vecteur de test

Il a été montré que le nombre de tests nécessaire et suffisant pour tester tous les courts-circuits d'un système à N connexions indépendantes est de $\log(N)$ (Kautz, 1974). Le test « *Counting Sequence Algorithm* » (ou « *Minimal-Size Test Set for shorts detection* »), génère un vecteur de test unique pour chaque connexion indépendante. Si le système est exempt de pannes, chaque réponse sera unique. La génération de la séquence peut être faite à l'aide d'un simple compteur. Dans le tableau 1.1, on observe que pour un nombre de 8 connexions indépendantes (N lignes horizontales), il faut $\log(N) = 3$ vecteurs.

Tableau 1.1 Séquence de vecteurs du test « *counting sequence algorithm* » pour un FPIN de 8 connexions indépendantes

| Connexion | Vecteurs de test parallèles | | |
|-----------|-----------------------------|----|----|
| | V1 | V2 | V3 |
| n1 | 0 | 0 | 0 |
| n2 | 0 | 0 | 1 |
| n3 | 0 | 1 | 0 |
| n4 | 0 | 1 | 1 |
| n5 | 1 | 0 | 0 |
| n6 | 1 | 0 | 1 |
| n7 | 1 | 1 | 0 |
| n8 | 1 | 1 | 1 |

Tableau 1.2 Séquence de vecteurs du test « *Modified counting sequence algorithm* » pour un FPIN de 8 connexions indépendantes

| Connexion | Vecteurs de test parallèles | | | |
|-----------|-----------------------------|----|----|----|
| | V1 | V2 | V3 | V4 |
| n1 | 0 | 0 | 0 | 1 |
| n2 | 0 | 0 | 1 | 0 |
| n3 | 0 | 0 | 1 | 1 |
| n4 | 0 | 1 | 0 | 0 |
| n5 | 0 | 1 | 0 | 1 |
| n6 | 0 | 1 | 1 | 0 |
| n7 | 0 | 1 | 1 | 1 |
| n8 | 1 | 0 | 0 | 0 |

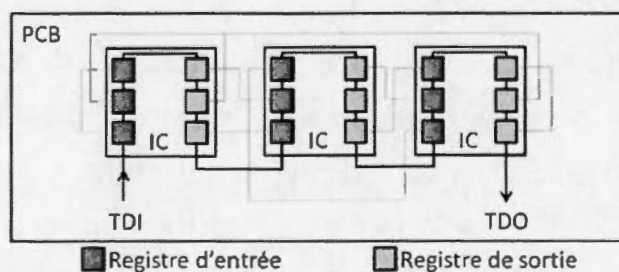


Figure 1.6 Topologie de réseau

Pour inclure le modèle S@, il faut que les vecteurs uniques de chaque connexion indépendante possèdent au moins un '1' et un '0'. Ainsi l'on se retrouve avec le test « *Modified Counting Sequence Algorithm* » (ou « *Minimal-Size Test Set for shorts and SA detection* ») dont le nombre de vecteurs de test est de $\log(N+2)$ (Tab. 1.2).

Si l'on suppose que le nombre de registres d'entrées est n et le nombre de registres de sorties est m , le réseau a $(n+m) = N$ registres (Fig. 1.6). Ainsi, pour le nombre de tests minimum $\log_2(N)$ de l'algorithme « *Counting Sequence Algorithm* », la complexité temporelle est $O(N \cdot \log_2(n))$. Pour le test « *Modified Counting Sequence Algorithm* », le nombre de tests est $\log_2(n+2)$, la complexité temporelle du test est $O(N \cdot \log_2(n+2))$.

Les deux tests précédents dépendent de l'architecture matérielle (c.-à-d. on différencie les entrées des sorties), alors, dans le cas d'une structure inconnue, un test indépendant de la structure est requis. Le test « *Order Independent test set* » reprend le test « *Modified Counting Sequence Algorithm* », mais effectue l'analyse sur les entrées et les sorties. Le nombre de tests devient $\log(N+2)$ et la complexité temporelle $O(N \cdot \log(N))$ (Hassan *et al.*, 1988).

Le problème des trois précédents tests est la sérialisation des tests, car l'on doit « pousser » la chaîne entière de test dans le registre JTAG. Afin de minimiser la taille des vecteurs de test, l'utilisation de l'algorithme « *Walking one* » (« *Walking zero* ») permet de générer un premier vecteur de la taille de la chaîne JTAG, puis de pousser un '0' (ou un '1') afin de passer au vecteur de test suivant. Cependant, la récupération du syndrome demande de séparer les vecteurs d'entrées des vecteurs de sorties, ou alors d'avoir un analyseur de syndrome in-situ. Sans avoir à pousser la chaîne entière des vecteurs de stimuli, on passe d'une complexité temporelle de $O(N \cdot \log_2(N))$ à $O(N)$. Un aspect important de l'algorithme « *Walking one* » est qu'il satisfait aux conditions afin de tester à la fois les S@ et les courts-circuits, de même que de garantir un diagnostic complet.

Il existe deux niveaux d'évaluation des pannes : le test et le diagnostic. Le test est la simple détermination de la présence ou non de pannes dans un système et le test de plusieurs pannes peut être effectué à l'aide d'un seul et même vecteur de test, car la localisation et caractérisation de ces pannes n'est pas nécessaire. Le diagnostic demande de déterminer la localisation de la panne et son type (Huang *et al.*, 1996; Yue et Dongfang, 2002), ainsi le nombre de vecteurs de test, de même que l'analyse des syndromes deviennent plus complexes. Et bien que d'autres tests fassent un compromis entre la minimisation des vecteurs de test et la couverture du diagnostic (*optimal algorithm, Equal-Weight & Min-Distance Algorithm* (Yang *et al.*, 2010)), dans ce mémoire, l'importance du diagnostic oblige à préférer l'algorithme « *Walking one* » qui permettra une couverture complète des S@ et des courts circuits.

Il est à noter que la génération des vecteurs de test de l'algorithme « *Walking one* » est assez simple pour pouvoir imaginer de le générer in situ sans demander trop de ressources, et ainsi réduire grandement le temps de test et diagnostic en supprimant le temps d'insertion des vecteurs de test.

1.4 Test et diagnostic des circuits logiques programmables

Les PCB ont des points communs avec les IoC et les FPGA utilisent les IoC. Ainsi, les FPGA de base ont un réseau d'interconnexions qui interconnecte des blocs logiques programmables (BL), par exemple nommés CLB (*Configurable Logic Block*) chez Xilinx ou LEs (*Logic Elements*) ou ALM (*Adaptive Logic Module*) chez Altera. De même, le réseau d'interconnexions de ces FPGA peut lui-même être divisé en deux parties, les interconnexions locales, qui relient les BL à l'IoC par le biais des CP (commutateur programmable, *crossbar*) et les interconnexions globales, qui relient tous les CP entre eux formant l'IoC. Dans ce mémoire, l'intérêt est porté sur les interconnexions globales, et plus particulièrement sur son test et diagnostic.

Il existe deux façons majeures d'effectuer un test et diagnostic (Doumar et Ito, 2003; Yue et Dongfang, 2002). Premièrement, la méthode que l'on nomme conception en

vue du test (*Design For Testability*, DFT) qui consiste à ajouter des ressources internes au design, comme des BIST (*built-in self-test*), pour permettre le test et diagnostic en interne du système sous test. Les BIST sont en général des registres à décalage à rétroaction (linéaire ou non) qui permettent de générer les vecteurs de test et d'en analyser la sortie. Une configuration du réseau est mise en place pour contrôler les commutateurs, puis les FSR (*feedback shift register*) sont mis en fonction. Cependant, en plus de la complexité de ces blocs, la logique de rétroaction est spécifique à un type de test et ne peut donc pas être réutilisée génériquement. De plus, ils demandent une surface non négligeable face à celle du FPIN sans DFT.

Pour résoudre ce problème, (Yue et Dongfang, 2002) ont mis le générateur de vecteurs de test in situ (*M-sequence generator*) alors que l'analyse des résultats se fait à l'extérieur (*External tester*) (Fig. 1.7). Ainsi cette méthode mixte permet de réduire le temps de téléchargement des vecteurs de test à un et de simplifier la logique de rétroaction du générateur de test. De plus, sa méthode garantit de pouvoir générer n'importe quelle configuration de test. Remarquons que cette configuration génère beaucoup de configurations, et qu'il faut les sélectionner à l'aide d'un registre de contrôle (*number setting register*, NSR).

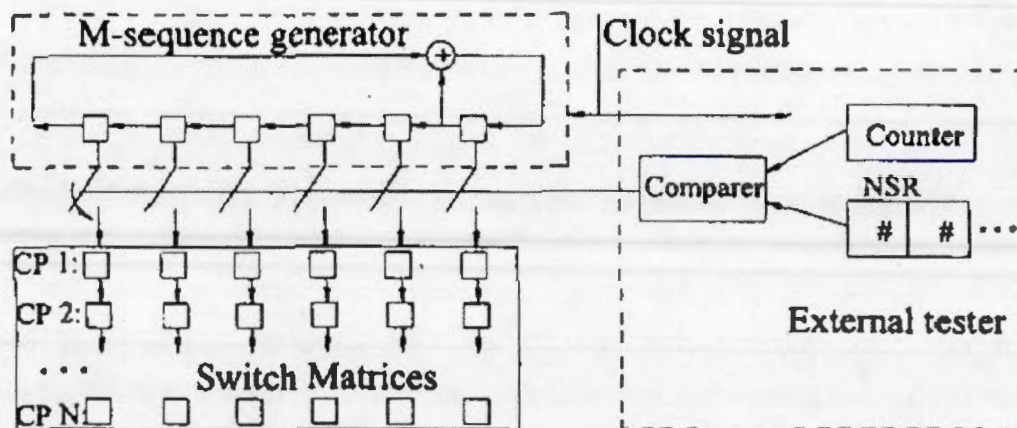


Figure 1.7 Système mixte de test interne/externe (Yue et Dongfang, 2002)

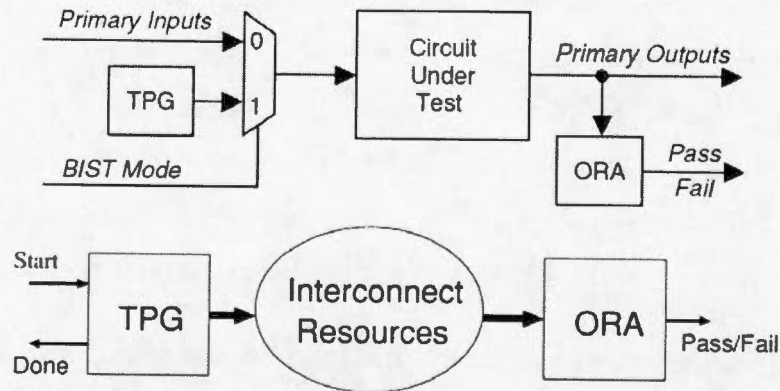


Figure 1.8 Structure de conception en vue du test (DFT) dans les circuits logiques programmables (Doumar et Ito, 2003; Wang *et al.*, 2006)

La seconde méthode est la génération et le traitement externe du test et diagnostic à l'aide de deux modules : un générateur de vecteurs de test (*test pattern generator*, TPG) qui génère les vecteurs de test, et d'un analyseur de syndromes (*output response analyser*, ORA) qui en analyse les résultats pour en faire l'évaluation souhaitée (Fig. 1.8).

Il est à noter que les termes TPG et ORA peuvent être aussi bien utilisés dans le cas de tests en externes que pour les tests internes.

Dans ces deux méthodes, l'étape consommant le plus de temps est le transfert des configurations du réseau vers le circuit sous test (*Design Under Test*, DUT). Ainsi, la majorité des travaux recherchent à réduire le nombre de configurations de test. Dans (Syng-Jyan et Chao-Neng, 1998), il faut onze configurations de test pour localiser une erreur unique. Dans (Yinlei et al., 1999a), cinq étapes sont nécessaire pour le modèle adaptatif de test, et huit étapes pour le test non adaptatif. Enfin, dans (Yinlei et al., 1999b) le nombre est réduit à six pour le diagnostic des pannes simples et de quelques pannes multiples. L'étude théorique de Renovell et al (Renovell *et al.*, 1997) montre que seulement trois configurations sont nécessaires pour tester tous les commutateurs programmables et lignes d'un réseau d'interconnexions pour les

pannes simples : Orthogonale, Diagonale-1 et Diagonale-2 (Fig. 1.9). Il est à noter que le modèle qu'ils utilisent est assez simplifié. Comme mentionné dans (Doumar et Ito, 2003), la minimisation du nombre de configurations est obtenue par la simplification du modèle, ce qui pénalise la couverture des pannes. Cependant, une adaptation de cette théorie à un réseau particulier est toujours possible (Renovell *et al.*, 1997; Renovell *et al.*, 1998) au prix d'un plus grand nombre de configurations.

On peut remarquer aussi que (Renovell *et al.*, 1997) simplifie son modèle en excluant l'utilisation de multiplexeurs. On peut retenir que le modèle utilisé pour la meilleure couverture possible doit être adapté à l'architecture étudiée (*custom*). De même, le FPIN de ce mémoire utilise un commutateur programmable à base de démultiplexeurs, ce qui n'est pas le cas des FPGA en général. Ainsi, le test du commutateur programmable demandera plus que trois configurations afin de couvrir la totalité des pannes possibles.

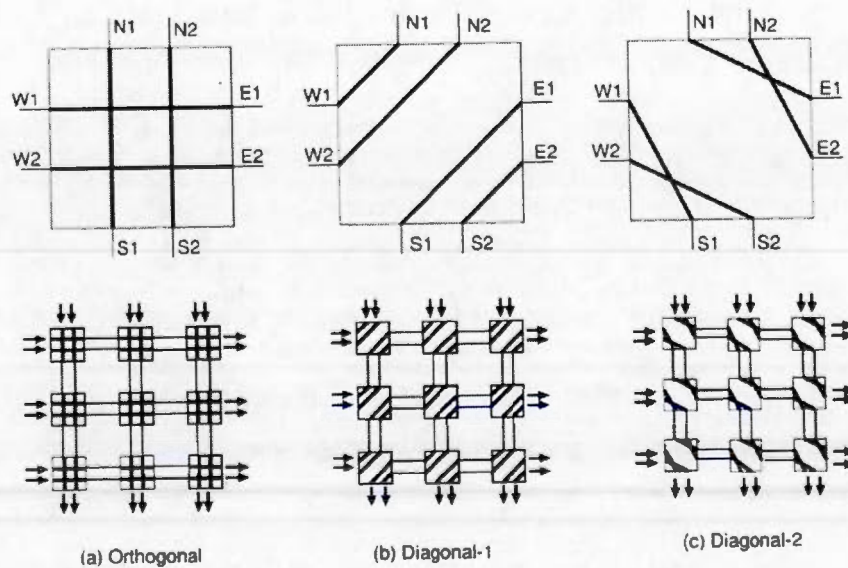


Figure 1.9 Configuration des commutateurs en vue du test des interconnexions (Doumar et Ito, 2003)

L'autre approche, celle d'ajouter de la logique pour générer les vecteurs de tests est une option déjà étudiée, comme dans (McCracken et Zilic, 2002). Pour les FPGA, il existe la possibilité d'utiliser les ressources internes pour faire des BIST (ce qui est différent du DFT), mais certains insèrent de la logique spécifique pour le test (DFT). La capacité des FPGA de pouvoir être programmée pour y mettre une logique particulière, et dans le cas de test et diagnostic, de pouvoir y programmer une partie de la logique pour en tester l'autre partie, permet bien des possibilités que les structures fixes comme le FPIN étudié dans ce mémoire n'ont pas. Cette différence représente une limite des comparaisons possibles du diagnostic du FPIN étudié dans ce mémoire avec le diagnostic des FPGA.

1.5 Résumé

De par sa taille, le FPIN étudié dans ce mémoire présente une probabilité de présence de pannes importante. Ainsi, l'utilisation d'une stratégie de tolérance aux pannes est primordiale pour atteindre un niveau de fiabilité lui permettant de fonctionner sans problème. Dans le chapitre 2 sera présentée l'architecture du FPIN, et plus particulièrement l'implémentation matérielle permettant la stratégie de tolérance aux pannes aussi bien pour le matériel fonctionnel, que pour le matériel de programmation/configuration. Le matériel de programmation/configuration du FPIN utilise un protocole JTAG, adapté de la norme IEEE-Std 1149.1, de chaînes programmable et tolérant aux pannes adapté, et utilise donc des chaînes de registres programmables pour transmettre et extraire les données du FPIN. Le diagnostic étant primordial pour contourner les pannes, la simple détection de ces pannes n'est pas suffisante, et l'optimisation temporelle des tests ne doit pas compromettre la diagnostabilité du FPIN. Ainsi, l'utilisation de l'algorithme « *Walking one* » semble être le plus adapté à la génération de série de vecteurs afin de permettre une couverture globale des pannes collé-à et court-circuit de ce circuit. De même, différents modèles de pannes ont été présentés dans ce chapitre, et le modèle le plus adéquat, comme celui utilisé dans (Basile-Bellavance, 2009), semble être le collé-à et

court-circuit. Tel qu'il sera présenté au chapitre 3, ce modèle dispose d'une abstraction permettant une simplicité d'application tout en gardant le diagnostic efficace.

CHAPITRE II

REVUE DE LA PLATEFORME DE PROTOTYPAGE RAPIDE

L'objectif principal de ce mémoire consiste à améliorer le test et diagnostic du FPIN dans le WaferIC™. Ce chapitre détaille la plateforme de prototypage rapide de système électronique WaferBoard™ qui est constituée de 4 structures dissociables. Les trois structures que sont le PCB supérieur (*top PCB*), le PCB inférieur (*bottom PCB*), et le module de puissance (*power supply*) seront discutées dans la première section. La dernière structure est l'interposeur intelligent, le WaferIC, et est constitué entre autres du réseau d'interconnexions programmable (FPIN). La plateforme dispose aussi, pour contrôler tout le matériel, d'une suite logicielle couvrant le flot de conception (*Workflow*) complet du système électronique en prototypage, permettant ainsi d'exploiter toutes les fonctionnalités du WaferBoard. Ce logiciel comprend entre autres un module pour le test et le diagnostic et sera donc brièvement présenté dans une deuxième section. Une troisième section présentera l'interposeur intelligent WaferIC dont le FPIN est une constituante. Le WaferIC utilise un protocole JTAG adapté, permettant la tolérance aux pannes, et qui permet la configuration ainsi que la communication avec le WaferIC et le système en développement. Ce protocole JTAG adapté, qui permet aussi la programmation et le test du FPIN, sera présenté dans la quatrième section. Enfin, dans une cinquième section, sera présentée en détail la structure principale d'étude de ce mémoire, le réseau d'interconnexions programmable FPIN. La revue détaillée de la plateforme de prototypage permettra de

comprendre l'environnement entourant le FPIN, de même que de décrire toutes les structures matérielles utilisées dans ce mémoire.

2.1 Plateforme de prototypage rapide de systèmes électroniques

Le projet de recherche DreamWafer™ visant à concevoir une plateforme de prototypage rapide de système électronique a été réalisé en coopération avec plusieurs universités canadiennes (Polytechnique Montréal, Université du Québec à Montréal, Université du Québec en Outaouais et McGill), de même que l'entreprise Gestion TechnoCap Inc. Cette plateforme de prototypage rapide permet d'interconnecter des composants électroniques discrets, comme des microprocesseurs/contrôleurs, des FPGA, de la mémoire, le tout en quelques heures au lieu de jours ou mois pour le prototypage traditionnel. Ces améliorations permettent à la fois une réduction des coûts et une arrivée plus tôt sur le marché (*Time To Market*, TTM), ainsi qu'une flexibilité de conception.

La plateforme matérielle, le WaferBoard, est composée de quatre principales parties : le *Top PCB*, le *Bottom PCB*, le module de puissance et le WaferIC (Fig. 2.1 et Fig. 2.2). Le *Top PCB* est le PCB supérieur de la plateforme qui gère la communication entre la plateforme et l'ordinateur utilisant la suite logicielle WaferConnect™, et permet ainsi la configuration et l'utilisation de la plateforme. Le *Bottom PCB* et le module de puissance permettent le contrôle de la puissance, et plus particulièrement celle fournie au WaferIC. Enfin le WaferIC, au cœur de la plateforme, est un réseau d'interconnexions programmable disposant à sa surface de plots, les NanoPads, permettant de détecter, d'alimenter ou de connecter à son réseau d'interconnexions programmable, les composants électroniques à sa surface.

Ainsi, il suffit de déposer les composants électroniques à interconnecter sur la surface du WaferIC, sans contrainte d'alignement (c.-à-d. que les composants électroniques peuvent être déposés de façon approximative sans avoir recours à un outil externe), et de fermer le couvercle pour assurer le contact entre les composants électroniques et le

WaferIC. Puis, à l'aide de la suite logicielle WaferConnect, la plateforme détecte les broches des composants électroniques pour ensuite configurer les interconnexions désirées à l'aide de son interface graphique ou d'un fichier de définitions des interconnexions (*Netlist*).

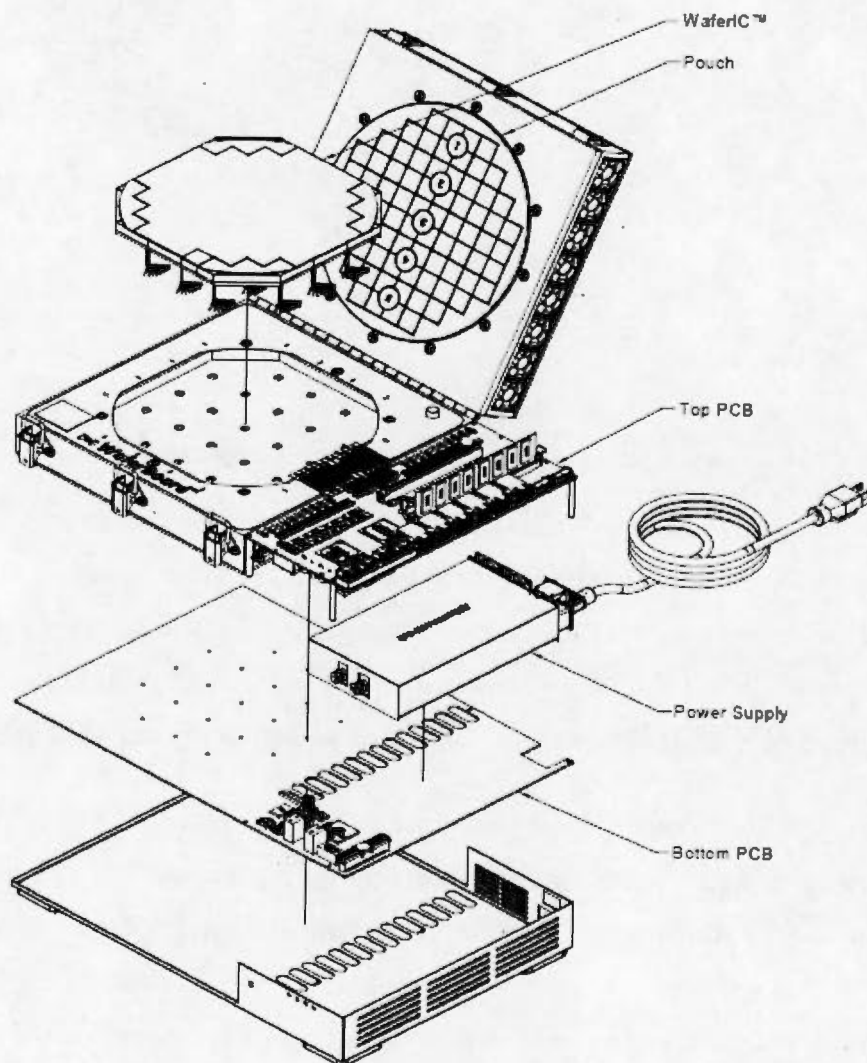


Figure 2.1 **Modèle éclaté de la plateforme de prototypage (André *et al.*, 2011)**



Figure 2.2 **Plateforme de prototypage rapide de systèmes électronique**
WaferBoard™

2.2 Suite logicielle de la plateforme de prototypage rapide

L'utilisation de la plateforme repose sur une suite logicielle permettant de tester, diagnostiquer, programmer et communiquer avec l'ordinateur de l'utilisateur. Le WaferConnect est le logiciel conçu pour gérer le flot de conception permettant la mise en œuvre des différentes étapes nécessaire à l'interconnexion des composants électroniques du système désiré. Ces étapes sont résumées dans la figure 2.3.

Une fois les composants électroniques déposés à la surface du WaferIC et le couvercle fermé, la plateforme de prototypage rapide peut être alimentée en tension. Suite à cette alimentation, la séquence de démarrage et de diagnostic de la plateforme s'effectue automatiquement. Le diagnostic permet de détecter et de localiser les pannes qui auraient pu être introduites lors de sa fabrication ou des utilisations précédentes ou par l'effet de l'usure, et ce diagnostic sera utilisé dans la suite du flot de conception, à l'étape du routage. Le diagnostic du FPIN étudié dans ce mémoire s'effectue à cette étape.

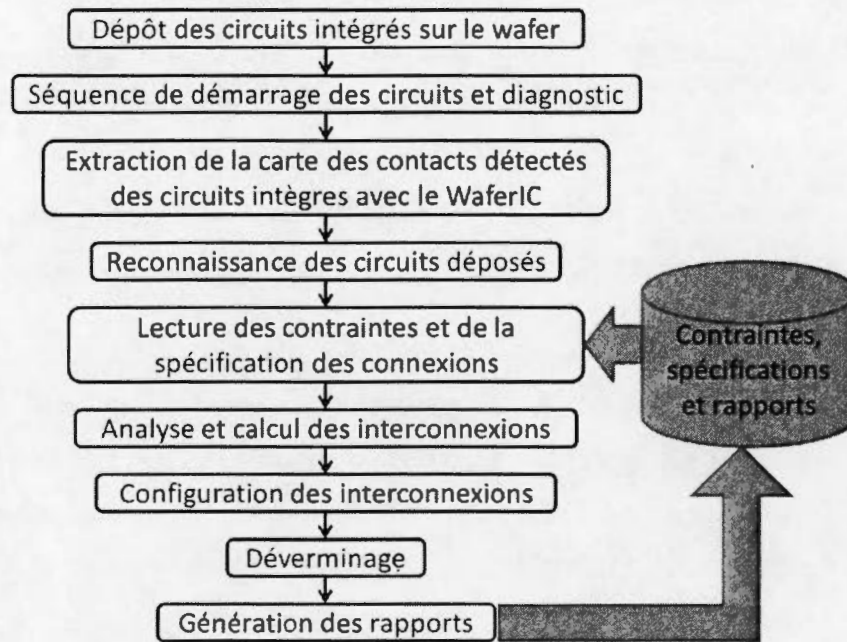


Figure 2.3 Flot de conception de la suite logicielle

Suite à cette séquence de démarrage, la suite logicielle lance une séquence de détection des broches des composants électroniques déposés à la surface du WaferIC. La détection est possible par la présence des NanoPads à la surface du WaferIC. Ces NanoPads sont assez petits pour que la présence d'une broche mette en court-circuit au moins deux de ces NanoPads. Chaque NanoPad dispose de la circuiterie nécessaire à la détection de court-circuit avec l'un de ses voisins. Ainsi, en réponse de la séquence de détection, la suite logicielle détermine la liste de tous les NanoPads en contact avec les broches des composants électroniques à la surface du WaferIC.

Cette liste de contacts est ensuite traitée pour pouvoir être comparée à une bibliothèque d'empreintes de composants électroniques (*Footprint*) et la suite logicielle fait une suggestion pour chaque composant électronique que l'utilisateur devra confirmer. Si aucune concordance n'est effectuée, l'utilisateur pourra entrer sa propre définition de composant pour poursuivre le flot de conception.

Une fois les composants identifiés, l'utilisateur fournira à la suite logicielle les spécifications des interconnexions désirées, de même qu'une liste de contraintes sur son système électronique en développement (fréquences de fonctionnement, alimentations des composants électroniques, longueurs de connexion, etc.). Cette entrée peut se faire à l'aide de fichiers d'une base de données, ou par l'interface graphique de la suite logicielle.

Enfin, la suite logicielle effectuera le routage des interconnexions et la programmation des NanoPads, en respectant les spécifications de l'utilisateur, de même que la liste des pannes connues et diagnostiquées au démarrage du système afin de les contourner, et configurera le WaferIC selon ce routage en utilisant un protocole JTAG adapté décrit dans la suite de ce chapitre. Pour finir, il générera une liste de rapports pour chacune des étapes précédentes.

Dans la section suivante, nous décrirons le cœur de la plateforme de prototypage rapide, le WaferIC.

2.3 Interposeur intelligent à l'échelle de la tranche de silicium

Le cœur de la plateforme de prototypage est le WaferIC, un composant actif sur lequel sont déposés, sans contrainte d'alignement, des composants électroniques à interconnecter. Le WaferIC est un circuit intégré à l'échelle de la tranche de silicium (*Wafer Scale Integrated Circuit*, WSIC), pouvant aussi être appelé circuit intégré de très grande surface (*Very Large Area Integrated Circuit*, VLAIC). Le WaferIC est une tranche de silicium de 200 mm de diamètre, dont la surface active, qui ne peut être photolithographiée en une seule fois, est la photorépétition de 76 réticules identiques. Au lieu de découper la tranche de silicium pour produire les puces électroniques, les réticules sont interconnectés par une technique de « couture » de réticules (*reticule stitching*) (Norman *et al.*, 2008). La technique de « couture » de réticules utilise de nouveaux masques de photolithographie pour déposer des couches de métallisations entre les réticules afin d'interconnecter les réticules voisins. Chaque

réticule de 17,28 mm de côté contient 1024 cellules (32×32) identiques à l'exception de l'état de leur mémoire de configuration, ce qui porte le nombre de cellules dans le WaferIC à 77 824. Cette surface du VLAIC, de même que la densité des interconnexions oblige donc au diagnostic du circuit de par la probabilité de pannes.

Chaque cellule de 540 μm de côté contient 16 contacts (4×4), appelés NanoPad, qui peuvent être programmés pour fournir aux *pins* (broches ou billes) du composant électronique en contact avec le NanoPad, une alimentation stable et régulée programmable, un contact à la masse, ou un lien vers le FPIN (Fig. 2.4 et Fig. 2.5). Chaque cellule dispose aussi d'un commutateur programmable de type *crossbar* lui permettant à la fois de connecter ses NanoPads au FPIN, mais aussi la configuration de ce FPIN. Les commutateurs programmables seront décrits en détail dans la section 2.5. Les 16 NanoPads par cellule portent le nombre à 1 245 184 NanoPads de 105 μm de côté sur toute la surface du WaferIC. Ce nombre et la taille de ces NanoPads permettent la détection, l'alimentation en tension et le contact à la masse ou au réseau d'interconnexions programmable de n'importe quel type de composant électronique, et ainsi le prototypage rapide de tous systèmes électroniques.

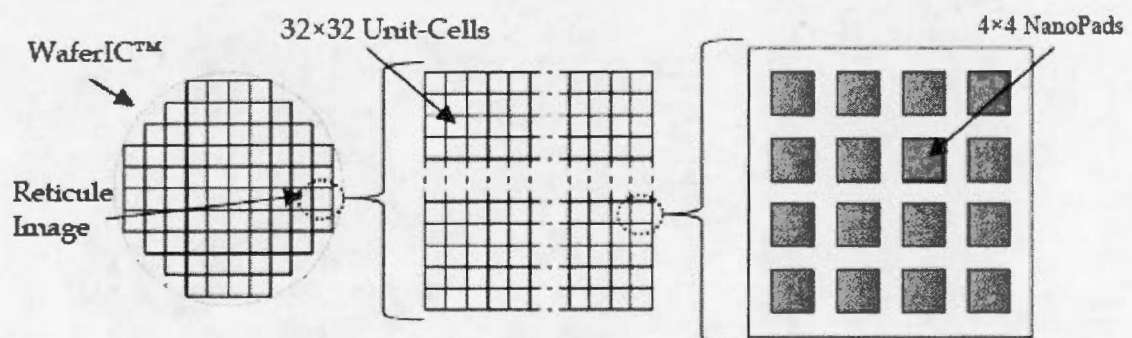


Figure 2.4 Architecture du réseau d'interconnexions programmable
(Norman *et al.*, 2008)

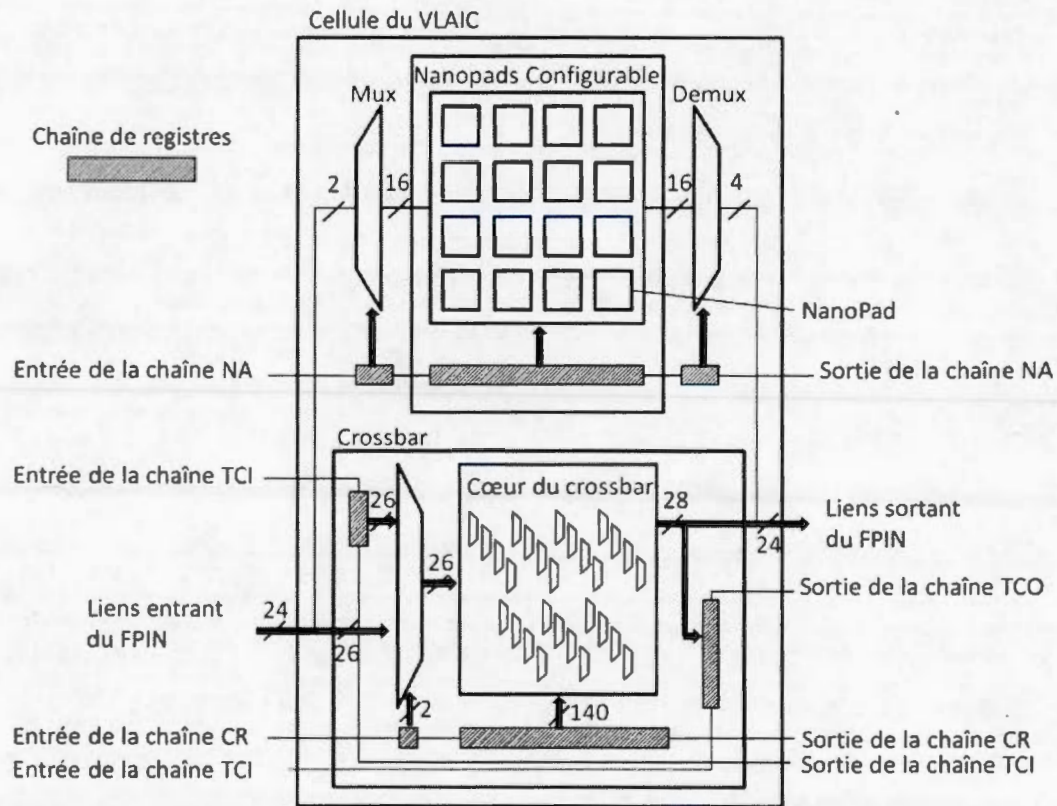


Figure 2.5 **Schéma d'une cellule du WaferIC™**

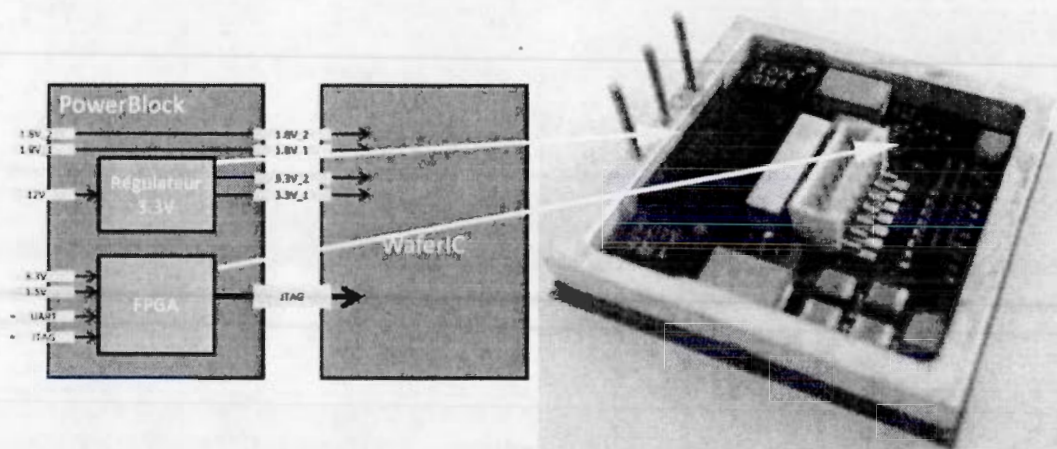


Figure 2.6 **Module WaferIC/PowerBlock, appelé MiniWaferIC**

Il est à noter que la surface du WaferIC est très sensible comme toutes les surfaces des tranches de silicium. Ainsi, un film anisotropique, appelé film en axe-Z (*Z-axis film*), est utilisé pour faire le lien conducteur selon l'axe des Z entre les NanoPads et les *pins* des composants électroniques.

Le WaferIC est connecté à la structure de la plateforme de prototypage à l'aide d'un module appelé PowerBlock. Cette connexion demande 21 modules PowerBlock connectés sous le WaferIC par des contacts (vias) traversant la tranche de silicium, les TSV (*Through Silicon Vias*). Chaque PowerBlock est un PCB permettant de gérer 4 réticules au travers 64 TSV par réticule (256 TSV par PowerBlock) et permettant l'alimentation en puissance (VDD1.8, VDD3.3 et GND), de même que la communication au travers 5 signaux JTAG par réticule (TCK, TRST, TMS, TDI et TDO) (Fig. 2.6).

Le module PowerBlock a été utilisé dans l'environnement de test afin d'appliquer le diagnostic proposé dans ce mémoire. Le module utilisé avec le MiniWaferIC (Fig. 2.6) seront présentés dans le chapitre 4, de même que son environnement de test.

Dans la section suivante sera présentée la version modifiée du protocole JTAG pour lui permettre de tolérer les pannes, et utilisée pour configurer et diagnostiquer le MiniWaferIC.

2.4 Système de configuration tolérant aux pannes

Le système de configuration du WaferIC, basé sur le protocole JTAG présenté à la section 1.3, est le moyen de configurer tous les composants de ce circuit intégré à l'échelle de la tranche de silicium. Ainsi, cette section présente ce protocole JTAG adapté pour la tolérance aux pannes et développé dans (Meng *et al.*, 2003) et (Basile-Bellavance, 2010).

Le WaferIC est constitué de réticules disposant tous d'une connexion JTAG permettant de les configurer en parallèle. Le protocole JTAG est utilisé pour configurer et diagnostiquer chaque cellule du réticule. Cette configuration et ce diagnostic se passent sur deux blocs distincts de la cellule, les NanoPads, dont les détails d'utilisation ne seront pas donnés dans ce mémoire, car ils en dépassent l'étude, et le commutateur programmable. Le commutateur programmable relie les Nanopads au réseau d'interconnexions. Les commutateurs programmables et leurs liens d'interconnexions forment le réseau d'interconnexions programmable FPIN. Les liens de ces commutateurs programmables relient les cellules voisines dans les quatre directions, Nord, Est, Sud et Ouest, distantes de 1, 2, 4, 8, 16 et 32 cellules.

Le protocole JTAG standard permettrait de disposer d'une seule chaîne à balayage définie et fixe, pour effectuer la configuration souhaitée en traversant toutes les cellules du réticule dans un ordre défini et fixe. Mais dû à la surface de ce circuit de la taille d'une tranche de silicium, des défauts sont inévitables et pourraient venir casser cette chaîne à balayage, rendant le réticule au complet inutilisable. Ainsi, il a été choisi d'appliquer la proposition de (Basile-Bellavance, 2010), soit d'utiliser une chaîne à balayage reconfigurable permettant d'éviter les cellules ou les liens JTAG défectueux afin de maximiser la possibilité de former une chaîne à balayage fonctionnelle. L'implémentation de la construction de route de chaîne JTAG au travers les cellules souhaitées a été implémenté dans ce mémoire afin de configurer et diagnostiquer le FPIN. De plus, le programme conçu a été utilisé dans les travaux de Safa Berrima (Berrima, 2015) pour appliquer l'algorithme de diagnostic de la chaîne JTAG tolérante aux pannes et reconfigurable.

Pour implémenter le protocole JTAG, chaque cellule du réticule dispose de son propre contrôleur TAP et de ses registres d'instructions et de données fonctionnant avec les mêmes spécifications que celles du protocole JTAG standard. L'utilisation des 5 signaux TDI, TDO, TMS, TCK et TRST permet de se déplacer dans la machine

à états finis présentée à la figure 2.7. Le signal TRST permet de revenir à l'état de départ qu'est « *test-Logic Reset* », de même qu'une suite de cinq '1' du signal TMS. Les transitions sont synchronisées par le signal TCK. Enfin, les déplacements dans les différents états de contrôleur TAP se font en fonction de la valeur du signal TMS à chaque cycle. Le principe étant de choisir le type de registre dont on souhaite avoir accès en se mettant dans l'état « *shift IR* » puis de pousser ou extraire la valeur du registre sélectionné en se mettant dans l'état « *shift DR* ».

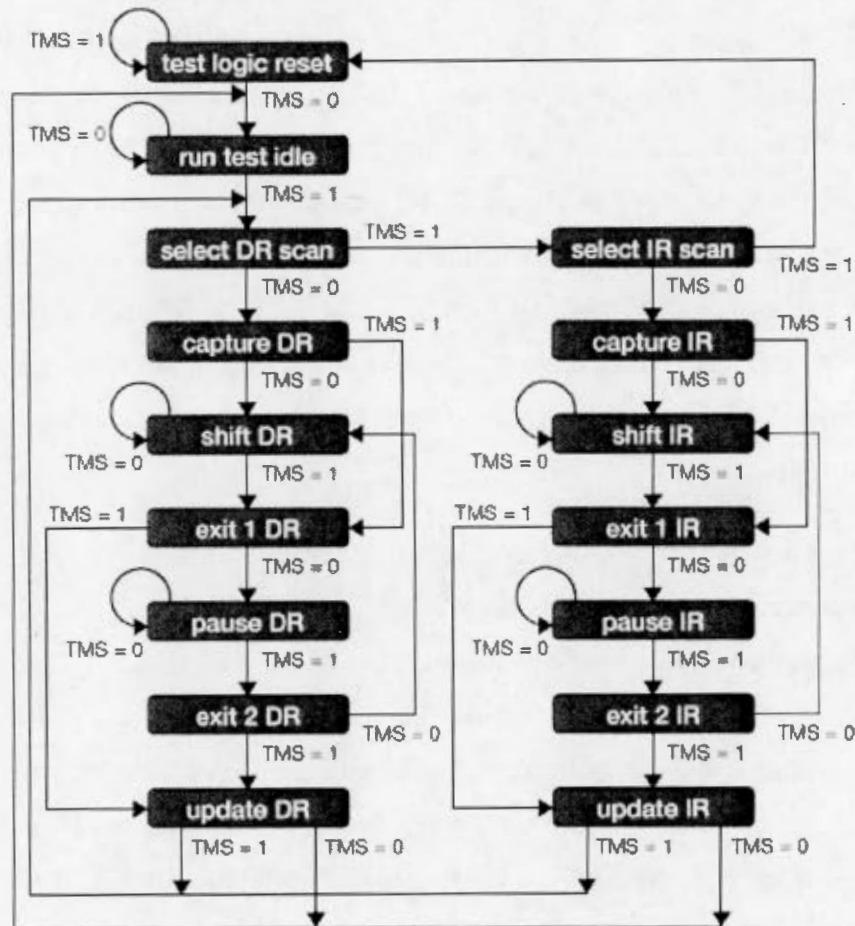


Figure 2.7 . Diagramme d'états du contrôleur TAP (www.xjtag.com/about-jtag/jtag-a-technical-overview/)

Ainsi pour écrire dans un registre spécifique, et partant de l'état '*test logic reset*', la séquence "01100" doit être envoyée pour arriver à l'état « *shift IR* », qui permet de pousser les bits de sélection du registre dont on veut l'accès. Cette chaîne permet de sélectionner les registres individuellement pour chaque cellule afin d'en configurer ou diagnostiquer une fonctionnalité particulière. Enfin, la commande "11100" permet d'arriver à l'état « *shift DR* », qui permet de pousser ou extraire la valeur du registre que l'on a sélectionné précédemment.

Cependant, à la différence de protocole JTAG standard, dans le WaferIC la chaîne JTAG n'est pas encore formée lors de la mise sous tension du circuit, et la première étape consiste à construire une chaîne JTAG parcourant les cellules dont on souhaite l'accès. Deux cellules dont la position est fixe (de par la structure matérielle) définissent le départ de la chaîne JTAG (entrée TDI) et sa fin (sortie TDO). Pour avoir une chaîne JTAG fonctionnelle, ces deux cellules doivent être connectées par un chemin valide afin de pouvoir extraire les bits de la chaîne JTAG poussés par la nouvelle séquence. Ainsi, comme le montre la figure 2.8, la création d'un chemin de la chaîne JTAG se passe en autant d'étapes que le nombre de cellules que forme la chaîne JTAG

La figure 2.8 représente une vue simplifiée d'un réseau de 4 par 4 cellules disposant d'une entrée JTAG (TDI) dans sa cellule en bas à gauche (1,1), et d'une sortie JTAG (TDO) sur la cellule directement au-dessus (1,2). La création d'une chaîne JTAG valide commence donc par la cellule (1,1) et doit se finir par la cellule (1,2). La plus petite chaîne JTAG possible peut donc être faite en programmant ces deux cellules. En commençant par programmer le registre destination de la cellule (1,1) vers le nord, puis en programmant les deux registres destination des deux cellules (1,1) et (1,2) dans les directions nord et ouest respectivement. La construction de la chaîne JTAG se fait par une concaténation de chaque direction des cellules par ajout d'une cellule à la fois.

À la figure 2.8, une chaîne plus complexe est présentée, mais le principe est identique. En partant de la cellule de départ de la chaîne JTAG (TDI), la programmation du premier lien vers la cellule suivante à l'Est est faite. Pour cela, la commande 'dest' (destination) est transmise au registre IR de la première cellule, puis la commande 'Est' à son registre DR. La première et deuxième cellule forme la nouvelle chaîne JTAG (Fig. 2.8 (1)). Pour la seconde étape, l'envoi d'une commandes 'dest' au registre IR, car le registre JTAG est maintenant de deux cellules, mais contient déjà la commande 'dest' de la cellule (1,1) qui sera alors poussée dans la cellule (2,1) par l'insertion de la nouvelle commande. Puis, nous envoyons les deux valeurs de 'dest' (Est.- Est) pour pointer la chaîne vers la troisième cellule (Fig. 2.8 (2)). Il est à remarquer que la valeur (DR) de la cellule la plus éloignée est envoyée en premier, puis la deuxième qui vient pousser la première. Une chaîne JTAG de 3 cellules est ainsi formée. La procédure est répétée, ainsi de suite jusqu'à former la chaîne désirée (Fig. 2.8 (3) à (8)), et qui se termine par la cellule TDO, permettant ainsi d'extraire les valeurs TDO vers l'extérieur.

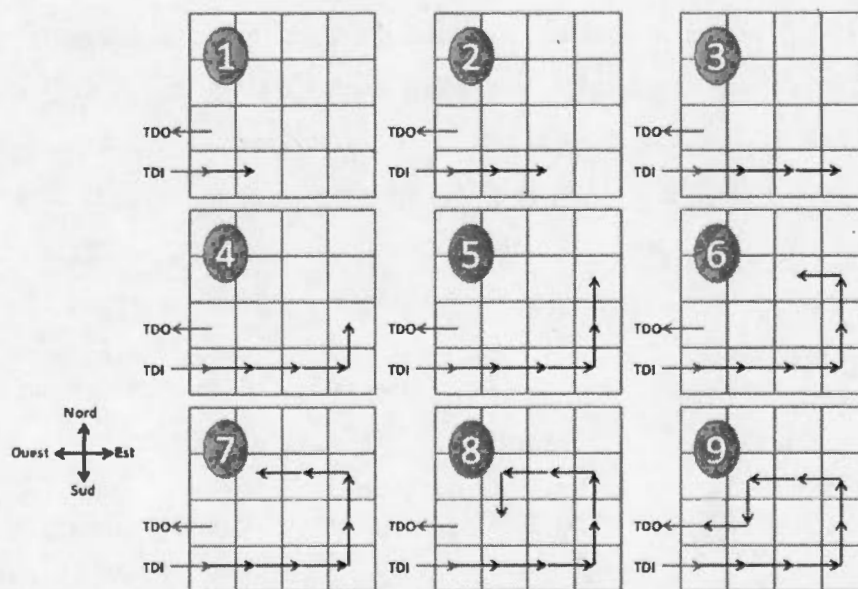


Figure 2.8 Construction d'une chaîne JTAG dans le FPIN

Le choix de l'implémentation matérielle de la chaîne JTAG tolérante aux pannes limite une chaîne à ne passer qu'une seule fois dans une cellule. Cette limitation pourrait être levée au prix d'une plus grande surface du contrôleur TAP.

Une autre fonctionnalité permettant la tolérance aux pannes est la possibilité de contrôler le TAP d'une cellule au travers du TAP d'une des quatre cellules voisines (Blaquiere *et al.*, 2014). Ainsi l'on retrouve, dans le tableau 2.1, l'accès aux 5 registres de configuration de la cellule (cd, na, cr, tci et tco) de même que ceux des 4 cellules voisines dans les directions, Nord, Est, Sud et Ouest (*_n, *_e, *_s et *_w).

Prenons le cas du réseau 4×4 de la figure 2.9, deux types de défectuosité peuvent avoir lieu, une défectuosité sur un lien (flèches rouges) ou sur un contrôleur TAP (bloc rouge). Ces deux défectuosités peuvent empêcher la formation de la chaîne JTAG entre les cellules d'entrée et de sortie (Fig. 2.9 (a)). Ainsi, la première possibilité pour contourner ces défectuosités consiste à former un lien passant par d'autres cellules (Fig. 2.9 (b)). La seconde est de programmer une cellule dont le contrôleur TAP est défectueux par l'intermédiaire d'une cellule voisine. Ainsi la cellule (4,3) ne pouvant pas être programmée directement, peut quand même l'être par les cellules voisines. De plus, le chemin ne passant pas par les cellules (4,1), (4,2), (4,3) et (4,4), ces dernières peuvent quant même être programmées par les cellules respectives (3,1), (3,2), (3,3) et (3,4). Ainsi, malgré les défectuosités présentes, le réseau 4×4 a pu trouver une chaîne JTAG qui permet de programmer toutes ses cellules.

Dans la prochaine section sera présentée la programmation du réseau d'interconnexions basé sur le protocole JTAG tolérant aux pannes.

2.5 Réseau d'interconnexions à l'échelle de la tranche de silicium

La section précédente a présenté la construction d'une chaîne JTAG au travers les cellules du réseau. Cette chaîne JTAG permet d'accéder aux registres de

configuration et de diagnostic des cellules. Trois types de registres sont utilisés dans la suite de ce mémoire, les registres CR (*CRossbar*), TCI (*Test Crossbar Input*) et TCO (*Test Crossbar Output*).

Le réseau d'interconnexions est composé de deux composantes majeures, les commutateurs programmables, qui sont la partie active du FPIN, et les liens qui les interconnectent. À la figure 2.10 est représentée une partie des liens (l'emphase étant sur les liens ouest) qui relie une cellule (en noir) aux autres. Ainsi le commutateur programmable compris dans la cellule est relié à 6 autres cellules dans chacune des quatre directions, Nord, Est, Sud et Ouest, et ceci dans les deux sens, entrant et sortant. Le tout formant $(6 \times 4) \times 2 = 48$ liens par cellule.

Chaque lien formant le réseau d'interconnexions relie un commutateur à celui d'une autre cellule. Le commutateur permet de relier les liens entrants de sa cellule aux liens sortants de cette cellule afin de former le routage désiré entre les liens entrants et sortants de cette cellule. De plus, des liens internes à la cellule permettent de connecter le bloc de Nanopads au FPIN.

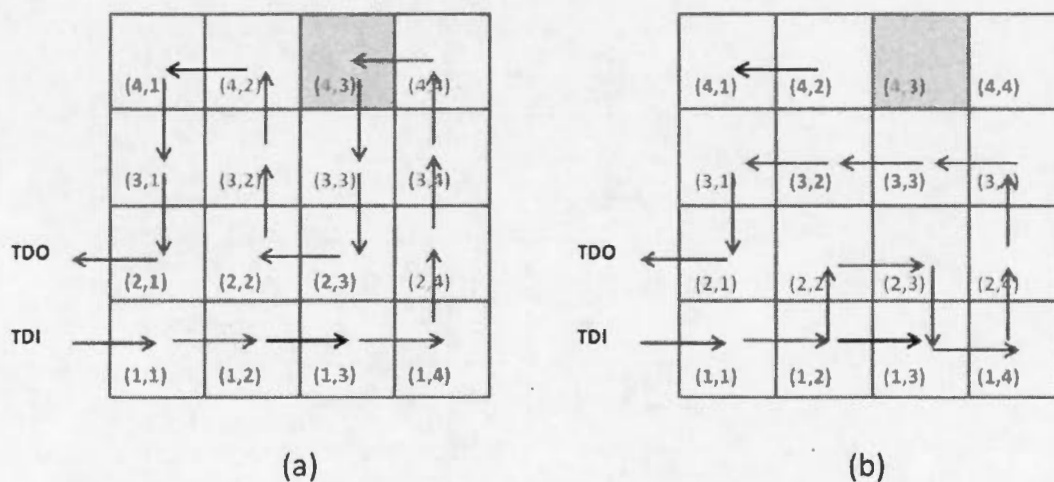


Figure 2.9 Construction d'une chaîne JTAG évitant les défauts

Tableau 2.1 Liste des commandes du registre d'instruction

| Registre d'instruction | | | | | Sortie activée |
|------------------------|-----|-----|-----|-----|----------------|
| (4) | (3) | (2) | (1) | (0) | |
| 0 | 0 | 0 | 0 | 0 | - |
| 0 | 0 | 0 | 0 | 1 | - |
| 0 | 0 | 0 | 1 | 0 | sc_dst_mode |
| 0 | 0 | 0 | 1 | 1 | sc_cd_mode |
| 0 | 0 | 1 | 0 | 0 | sc_na_mode |
| 0 | 0 | 1 | 0 | 1 | sc_cr_mode |
| 0 | 0 | 1 | 1 | 0 | sc_tci_mode |
| 0 | 0 | 1 | 1 | 1 | sc_cd_n_mode |
| 0 | 1 | 0 | 0 | 0 | sc_cd_s_mode |
| 0 | 1 | 0 | 0 | 1 | sc_cd_w_mode |
| 0 | 1 | 0 | 1 | 0 | sc_cd_e_mode |
| 0 | 1 | 0 | 1 | 1 | sc_na_n_mode |
| 0 | 1 | 1 | 0 | 0 | sc_na_s_mode |
| 0 | 1 | 1 | 0 | 1 | sc_na_w_mode |
| 0 | 1 | 1 | 1 | 0 | sc_na_e_mode |
| 0 | 1 | 1 | 1 | 1 | sc_cr_n_mode |
| 1 | 0 | 0 | 0 | 0 | sc_cr_s_mode |
| 1 | 0 | 0 | 0 | 1 | sc_cr_w_mode |
| 1 | 0 | 0 | 1 | 0 | sc_cr_e_mode |
| 1 | 0 | 0 | 1 | 1 | sc_tci_n_mode |
| 1 | 0 | 1 | 0 | 0 | sc_tci_s_mode |
| 1 | 0 | 1 | 0 | 1 | sc_tci_w_mode |
| 1 | 0 | 1 | 1 | 0 | sc_tci_e_mode |
| 1 | 0 | 1 | 1 | 1 | - |
| 1 | 1 | 0 | 0 | 0 | - |
| 1 | 1 | 0 | 0 | 1 | sc_tco_n_mode |
| 1 | 1 | 0 | 1 | 0 | sc_tco_s_mode |
| 1 | 1 | 0 | 1 | 1 | sc_tco_w_mode |
| 1 | 1 | 1 | 0 | 0 | sc_tco_e_mode |
| 1 | 1 | 1 | 0 | 1 | - |
| 1 | 1 | 1 | 1 | 0 | sc_tco_mode |
| 1 | 1 | 1 | 1 | 1 | bypass_mode |

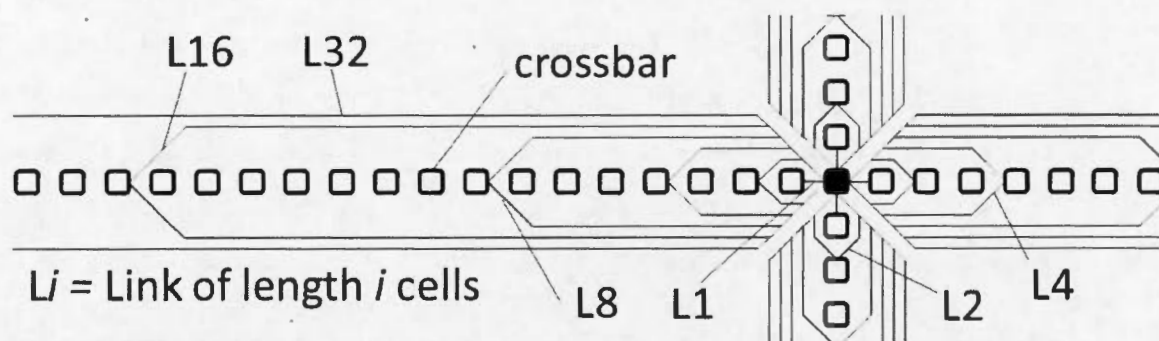


Figure 2.10 Interconnexions d'une cellule dans le FPIN

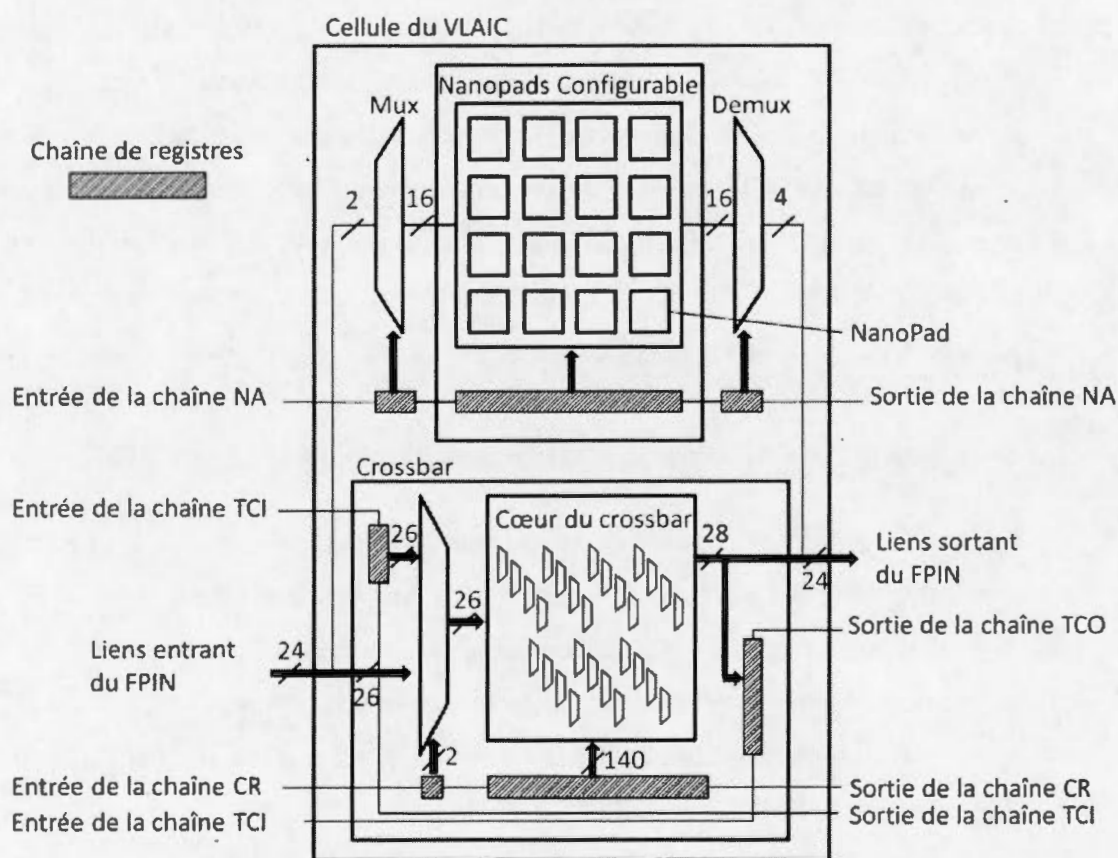


Figure 2.11 Schéma d'une cellule du WaferIC™

Le routage du commutateur s'effectue principalement dans le cœur du commutateur formé de 26 démultiplexeurs, chaque entrée du commutateur étant connecté à un démultiplexeur. Aux 24 liens entrants de la cellule s'ajoutent deux liens internes provenant du bloc des Nanopads. De même, aux 24 sorties nécessaires pour les liens sortants de la cellule, 4 sorties supplémentaires du commutateur le relient au bloc des Nanopads. Ces 26 démultiplexeurs disposent donc de 28 sorties connectées aux 24 liens sortant de la cellule auxquels s'ajoutent les 4 liens internes vers les Nanopad.

Le cœur du commutateur est contrôlé par le registre CR de la cellule, composé de 26 blocs de 5 bits, nécessaires à la sélection de la sortie de chacun des 26 démultiplexeurs, auxquels s'ajoutent 2 blocs de 2 bits, l'un au début, l'autre à la fin du registre CR. Ces 4 bits sont des bits nécessaires au test. Le plus important étant le bit contrôlant l'entrée du cœur du commutateur et permettant de choisir entre le mode normal de fonctionnement qu'est la connexion au réseau d'interconnexions et le mode test qui relie l'entrée du commutateur aux registres TCI d'insertions de vecteur de taille 26 bits. Le mode test permet, entre autres, l'injection de vecteur de test afin de diagnostiquer les commutateurs et les liens d'interconnexions. Enfin, un registre permet la lecture de l'état des sorties du Crossbar sans avoir à altérer le réseau d'interconnexions au contraire du registre d'insertion TCI qui déconnecte le commutateur des liens entrants. Ce registre de 28 bits est le registre TCO.

L'abondance des interconnexions entre les cellules permet d'offrir des alternatives de connexion entre deux cellules si le lien le plus direct est défectueux, et ainsi permettre une tolérance aux pannes. Cependant, pour permettre une tolérance aux pannes, une connaissance de l'état de tous les commutateurs et liens du système est fondamentale. Ainsi, nous présentons dans ce mémoire les techniques mises en place pour tester et diagnostiquer le FPIN.

2.6 Résumé

Ce chapitre a détaillé tous les composants du WaferIC nécessaires à la fois pour la configuration des chaînes JTAG et pour la configuration des cellules permettant d'appliquer un algorithme de test et diagnostic.

La construction et configuration de la chaîne JTAG présentée dans ce chapitre sont fondamentales pour l'évaluation des performances de l'algorithme de diagnostic présentée dans le présent mémoire. Elles sont à la base de l'outil logiciel permettant l'application de l'algorithme.

La structure des registres de configuration de la cellule a été présentée, car elle est nécessaire à la compréhension de l'application de l'algorithme de diagnostic des commutateurs et des liens présenté au chapitre 3.

CHAPITRE III

TEST ET DIAGNOSTIC DU RÉSEAU D'INTERCONNEXIONS PROGRAMMABLE

Le diagnostic d'un circuit intégré est la localisation des pannes, ou à l'inverse, des éléments fonctionnels de ce circuit. Dans le cas du réseau d'interconnexions programmable à l'échelle de la tranche de silicium présenté dans ce mémoire, le diagnostic prend une importance cruciale, car la surface de 200 mm^2 de ce circuit intégré augmente grandement la probabilité de présence de pannes. De plus, l'utilisation particulière du circuit, présenté à la section 2.5, vient ajouter une probabilité de présence de pannes par la répétition des dépôts de composants électroniques à sa surface qui n'est simplement recouverte que d'une fine couche de conducteur anisotropique. Cette manipulation peut engendrer de nouvelles pannes, tout comme l'usure même du circuit ou la présence de poussières.

Le diagnostic est utilisé dans le réseau d'interconnexions programmable (*Field Programmable Interconnection Network*, FPIN) afin de dresser une carte des régions défectueuses, permettant l'application d'une stratégie de tolérance aux pannes par le contournement de ces pannes. Un diagnostic à chaque utilisation est donc nécessaire, et le temps de ce diagnostic devient une contrainte importante par le coût qu'il entraîne. Ainsi, ce chapitre propose un algorithme permettant de réduire jusqu'à 112 fois le temps de diagnostic des commutateurs programmables et des liens d'interconnexions du FPIN par la couverture des pannes de type collé-à et court-circuit, par rapport à l'algorithme précédemment proposé par (Basile-Bellavance, 2009). Une première section présentera l'algorithme proposé pour le diagnostic des

commutateurs programmables, alors qu'une seconde section présentera l'algorithme proposé pour le diagnostic des liens d'interconnexions. Enfin, une troisième section présentera une analyse des temps de test et de la complexité de l'algorithme proposé en comparaison de l'algorithme présenté dans (Basile-Bellavance, 2009).

3.1 Algorithme de diagnostic des commutateurs dans le réseau d'interconnexions programmable

Chaque cellule d'un réseau dispose d'un réseau de commutateurs programmables (*crossbar*) et des registres permettant à la fois la configuration et le test de ce commutateur. Ces ressources permettent d'effectuer localement le test du commutateur, et donc de paralléliser le test au travers tout le FPIN.

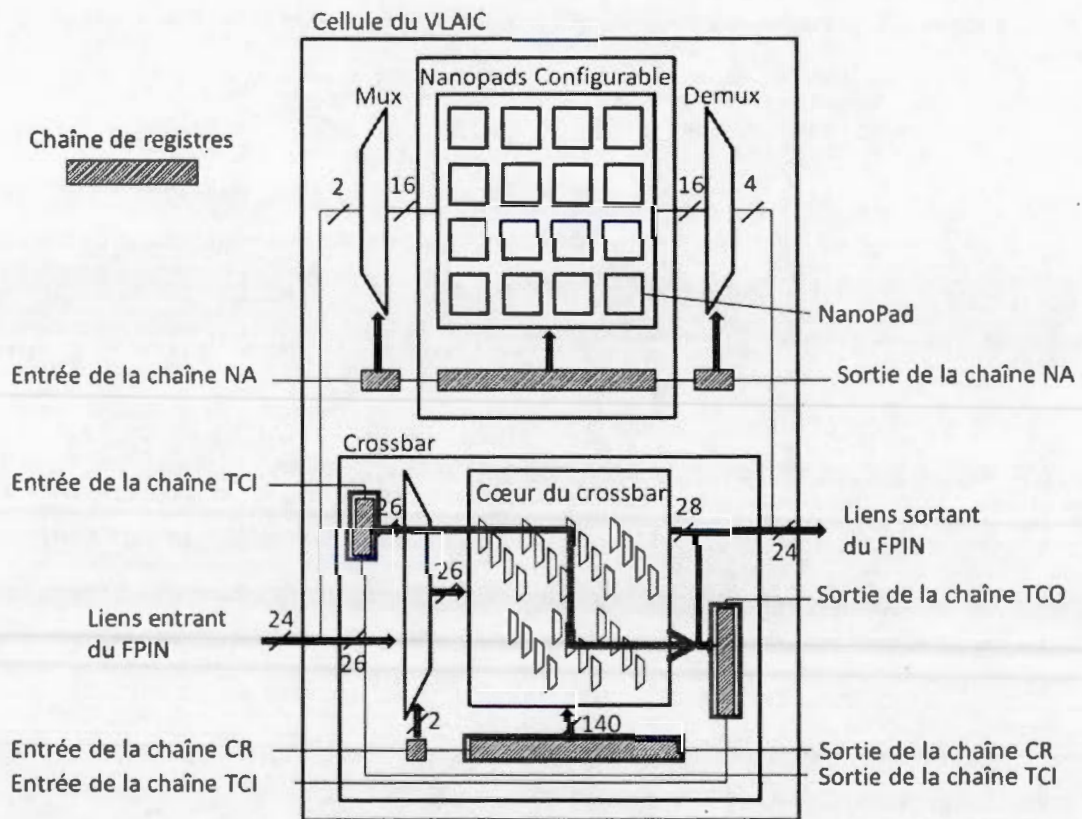


Figure 3.1 Cellule du VLAIC et ses chaînes de registres

Chaque réticule dispose d'une chaîne JTAG permettant l'accès aux registres de configuration (CR), d'insertion de vecteurs de tests (TCI) et d'extraction de syndromes (TCO) du commutateur (Fig. 3.1). Cette chaîne JTAG, présentée à la section 2.4, est programmable pour permettre de contourner les registres défectueux. Afin de simplifier l'explication, le cas d'une chaîne passant par toutes les cellules d'un réticule est pris en compte.

L'algorithme de diagnostic du commutateur proposé contient trois étapes principales (Fig. 3.2 et Fig. 3.3) :

1. **La première étape** du diagnostic est la configuration des commutateurs en mode *broadcast* par les registres de configuration (CR) (Fig. 3.3). Ce mode *broadcast* interconnecte une entrée du commutateur à toutes ses sorties. Le test devra ainsi être répété pour chaque entrée par une nouvelle configuration du commutateur en mode *broadcast*, pour en tester toutes les entrées.
2. **La seconde étape** est l'insertion du vecteur de test par le registre TCI. Le vecteur de test est composé d'un '1' logique sur l'entrée sous test du commutateur alors que les autres entrées sont à '0'. Il est alors possible d'observer les courts-circuits de type '0' logique dominant (*AND bridge*) des entrées non testées sur l'entrée sous test ou sur les sorties du commutateur. De même, il est possible d'observer les collages à '0' (*S@0*) pour cette entrée sous test, ainsi que pour toutes les sorties du commutateur.
3. **La troisième étape** du test est la récupération du syndrome de test par le registre TCO.

Afin de compléter la couverture du test, les étapes 2 et 3 sont répétées pour le vecteur de test complémentaire :

1. **La seconde étape complémentaire** (étape 2') est l'insertion du vecteur de test par le registre TCI. Le vecteur de test est composé d'un '0' logique sur

l'entrée sous test du commutateur alors que les autres entrées sont à '1'. Il est alors possible d'observer les courts-circuits de type '1' logique dominant (*OR bridge*) des entrées non testées sur l'entrée sous test ou sur les sorties du commutateur. De même, il est possible d'observer les collages à '1' (*S@1*) pour cette entrée sous test, ainsi que pour toutes les sorties du commutateur.

2. La troisième étape complémentaire (étape 3') du test est la récupération du syndrome de test par le registre TCO.

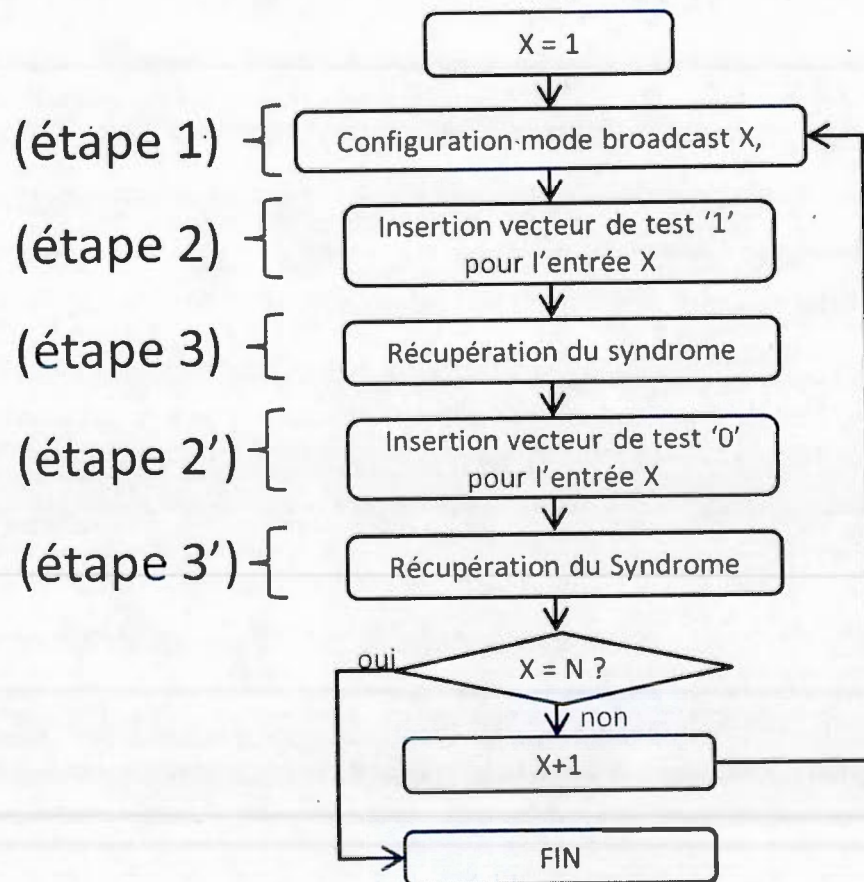


Figure 3.2 Algorithme de diagnostic d'un *crossbar* à N entrées

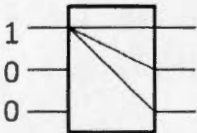
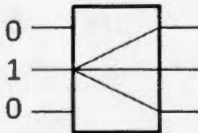
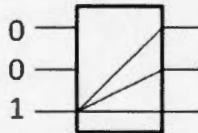
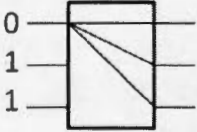
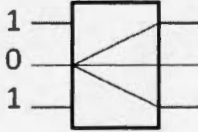
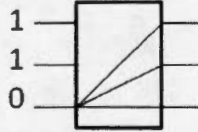
| Configuration <i>broadcast #1</i> | Configuration <i>broadcast #2</i> | Configuration <i>broadcast #3</i> | |
|---|---|--|-----------------------------------|
|  |  |  | Premier vecteur de test |
|  |  |  | Vecteur de test complémentaire |

Figure 3.3 Configurations et vecteurs de test pour un *crossbar* à
N=3 entrées

L'algorithme décrit ici est en principe le même que celui proposé par (Basile-Bellavance, 2009), et seule l'application le différencie de par l'architecture différente des registres de configuration des commutateurs, de même que des registres d'insertion et d'extraction des tests. Alors que l'architecture actuelle dispose des registres de configuration CR (142 bits), d'insertion de vecteurs de test TCI (26 bits) et d'extraction de syndromes TCO (28 bits), l'architecture précédente dispose de 4 registres regroupés en deux chaînes, l'un pour la configuration (CR et NA réunie en 314 bits), et l'autre pour le test (TCI et TCO réunie en 61 bits). Ainsi, l'insertion des vecteurs de test, et l'extraction des syndromes se font en même temps sur l'architecture précédente, alors qu'ils se font en deux temps dans l'algorithme proposé ici. Dans l'architecture précédente, l'insertion des vecteurs de test pousse la chaîne contenant les syndromes du test précédent, permettant leur extraction. Il n'y a donc aucun gain de temps significatif sur l'insertion des vecteurs de test et l'extraction des syndromes. Cependant, la programmation des commutateurs qui demande 314 bits par commutateur dans l'architecture précédente, n'en demandent

plus que 142 bits dans l'architecture utilisée dans ce mémoire, soit une réduction de 55 % du temps de configuration des commutateurs programmables.

La figure 3.4 présente le modèle de pannes uniques d'un commutateur à $N=3$ entrées et 3 sorties. Les pannes testées peuvent être un collage à l'entrée (b) ou à la sortie (c) du commutateur, de même qu'un court-circuit entre deux entrées (d), deux sorties (e), ou une entrée et une sortie (f). La colonne (a) de la figure 3.4 représente un commutateur non défectueux, et les lignes (C1), (C2), et (C3) sont les trois configurations possibles d'un *crossbar* 3×3 en mode *broadcast*. Le tableau 3.1 présente les $2N$ syndromes (avec $N=3$, le nombre d'entrée du *crossbar*) extraits du test appliqué aux types de pannes de la figure 3.4, à l'exception des courts-circuits entre les sorties, qui ne peuvent pas être diagnostiqués par l'algorithme de diagnostic des *crossbar* mais le sont par l'algorithme de diagnostic des liens présenté à la section suivante.

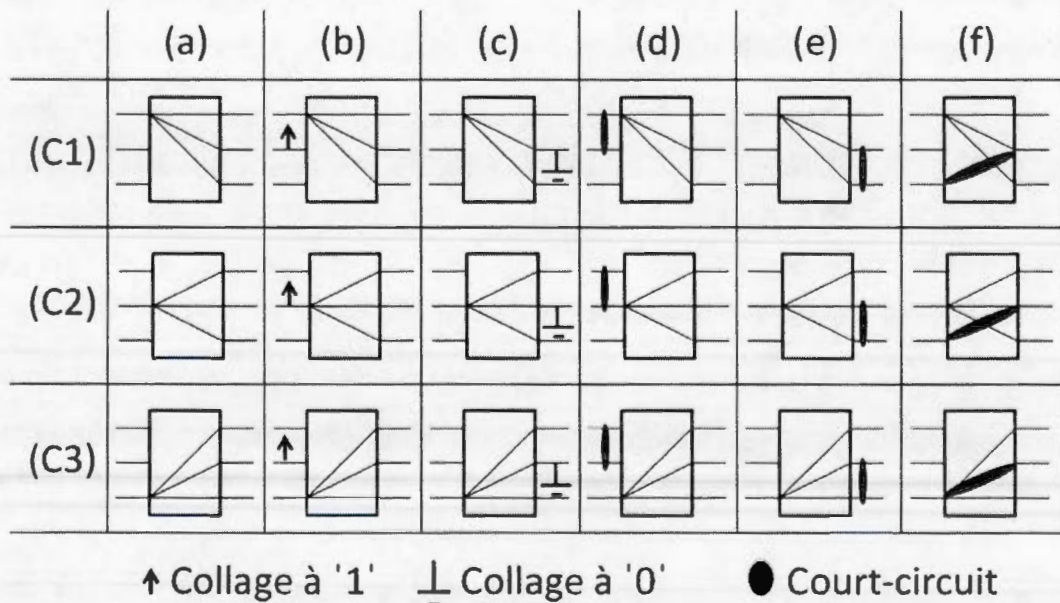


Figure 3.4 **Modèle de pannes d'un *crossbar* 3×3**

Tableau 3.1 Syndrome produit par des pannes simples diagnosticables pour un *crossbar* 3×3

| Vecteur de test | (a) syndromes | (b) syndromes | (c) syndromes |
|-----------------|---------------|---------------|---------------|
| 1 0 0 0 1 1 | 1 1 1 0 0 0 | 1 1 1 0 0 0 | 1 0 1 0 0 0 |
| 0 1 0 1 0 1 | 1 1 1 0 0 0 | 1 1 1 1 1 1 | 1 0 1 0 0 0 |
| 0 0 1 1 1 0 | 1 1 1 0 0 0 | 1 1 1 0 0 0 | 1 0 1 0 0 0 |

| (d) Syndromes OR bridge | (d) Syndromes AND bridge | (f) Syndromes OR bridge | (f) Syndromes AND bridge |
|----------------------------|-----------------------------|----------------------------|-----------------------------|
| 1 1 1 1 1 1 | 0 0 0 0 0 0 | 1 1 1 0 1 0 | 1 0 1 0 0 0 |
| 1 1 1 1 1 1 | 0 0 0 0 0 0 | 1 1 1 0 1 0 | 1 0 1 0 0 0 |
| 1 1 1 0 0 0 | 1 1 1 0 0 0 | 1 1 1 0 0 0 | 1 1 1 0 0 0 |

Les différents syndromes du tableau 3.1 montrent comment les différents types de pannes peuvent être diagnostiqués. Chaque ligne du tableau 3.1 représente une configuration en mode *broadcast* ((C1), (C2) et (C3) de la Fig. 3.4) et ses vecteurs de test et syndromes correspondants.

Par exemple, si le test est appliqué sur un commutateur dont une entrée est S@'1' (Fig. 3.4 colonne (b)), les deux vecteurs de test appliqués à la configuration en mode *broadcast* pour l'entrée 1 du commutateur (C1) (Fig. 3.3 colonne 1) correspond à la première ligne de la case « vecteur de test » (Tab. 3.1). Les syndromes résultants sont la première ligne de la colonne (b) du tableau 3.1.

La comparaison des syndromes d'un type de panne avec ceux d'un commutateur non défectueux (Tab. 3.1 colonne (a)) permet d'extraire les signatures caractéristiques de ces types de pannes.

Ainsi, le S@'1' de la figure 3.4 colonne (b) est diagnostiqué par le vecteur de test "010" appliqué à la configuration en mode *broadcast* de la deuxième entrée du crossbar (C2). Le syndrome généré par cette panne est "111" (Tab. 3.1 colonne (b) deuxième ligne, deuxième vecteur) au contraire du syndrome d'un commutateur non défectueux "000" (tab. 3.1 colonne (a) deuxième ligne, deuxième vecteur). Les empreintes des pannes sur les syndromes ont été ombragées dans le tableau 3.1.

Cet exemple simple montre que l'algorithme proposé de diagnostic des commutateurs programmables permet de détecter les collages et les courts-circuits du commutateur, à l'exception des courts-circuits entre ses sorties couvertes par l'algorithme de diagnostic des liens du FPIN. De plus, l'algorithme permet de différencier clairement ces pannes par l'empreinte qu'elles forment sur les syndromes (Tab. 3.1).

3.2 Algorithme de diagnostic des liens du FPIN

Le diagnostic du FPIN est divisé en deux étapes, où les crossbars sont dans un premier temps traités (Sec. 3.1) suivi du test des liens d'interconnexion. Les liens d'interconnexions du FPIN ont des longueurs de 1, 2, 4, 8, 16 ou 32 cellules, et les tests sur deux liens de deux cellules d'une même ligne du réseau (horizontale ou verticale), et distants de moins de 32 cellules si ces liens vont dans la même direction, risquent d'interférer l'un avec l'autre. Si les directions de ces liens sont opposées, la distance passe de 32 à 64 cellules de distance. Ceci oblige la définition d'une région du FPIN où seul un test de diagnostic des liens peut être effectué à la fois. Cette zone est le cône d'influence du test.

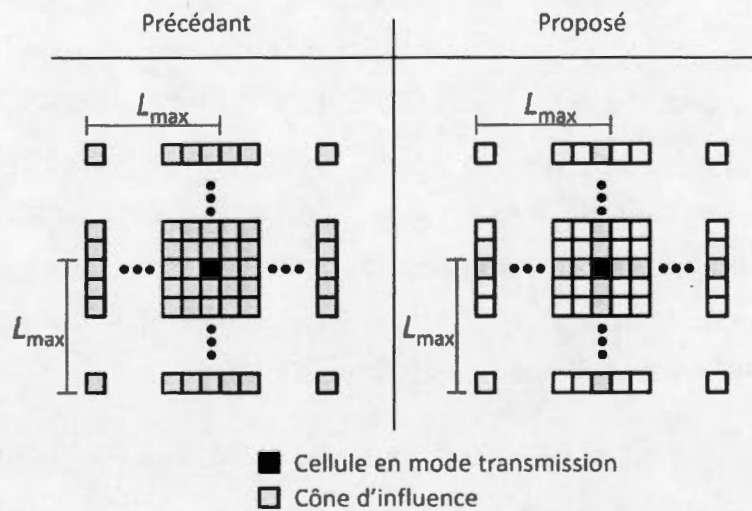


Figure 3.5 Cône d'influence des deux algorithmes

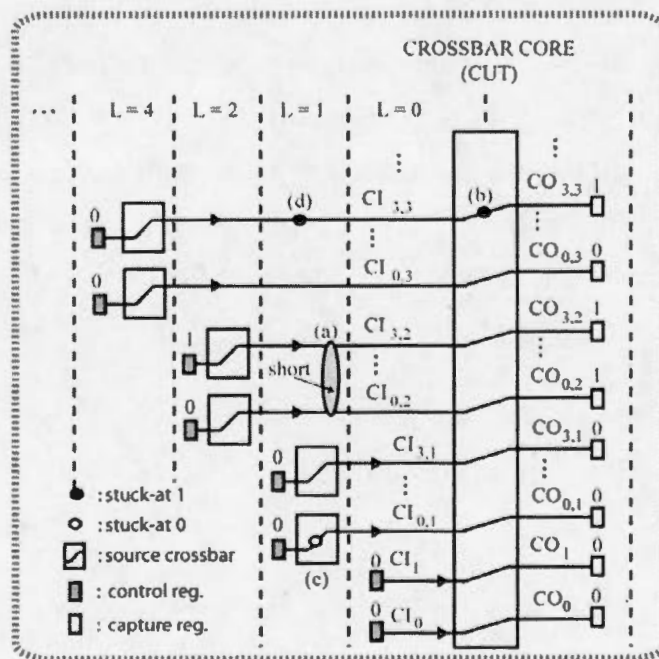


Figure 3.6 Algorithme précédemment proposé pour le test des liens
(Basile-Bellavance, 2009)

Dans l'algorithme proposé par (Basile-Bellavance, 2009), le cône d'influence est complexe et large de 64×64 cellules (Fig. 3.5) afin d'éviter toute interférence entre les tests. L'algorithme de diagnostic des liens du FPIN proposé dans ce mémoire applique le même algorithme de test « *Walking one* », et son complément, aux liens d'interconnexions entre les *crossbars*, comme proposé dans (Basile-Bellavance, 2009). Cependant, en tirant avantage de l'architecture du réseau d'interconnexions, l'algorithme proposé dans ce mémoire réduit le cône d'influence afin de permettre une diagonalisation du diagnostic des liens du FPIN.

Les liens d'une cellule sont soit horizontaux, soit verticaux. Ainsi dans l'algorithme proposé, le cône d'influence est réduit aux seules cellules de la même ligne, horizontale et verticale, distantes de moins de 32 cellules (Fig. 3.5). Ce cône d'influence permet la diagonalisation du test comme présenté à la figure 3.7. De plus, il permet de tester tous les réticules parallèlement avec la même configuration et les mêmes vecteurs de test. Cette diagonalisation permet de tester N cellules en parallèle pour chacun des réticules de $N \times N$ cellules. Lorsque le test de la première diagonale est accompli, le test passe à la diagonale suivante, comme le montre la figure 3.7. Le premier test demande la configuration de toutes les cellules du réticule, mais les tests suivants ne demandent, chacun, que de reconfigurer 2 diagonales afin de faire avancer la diagonale de test.

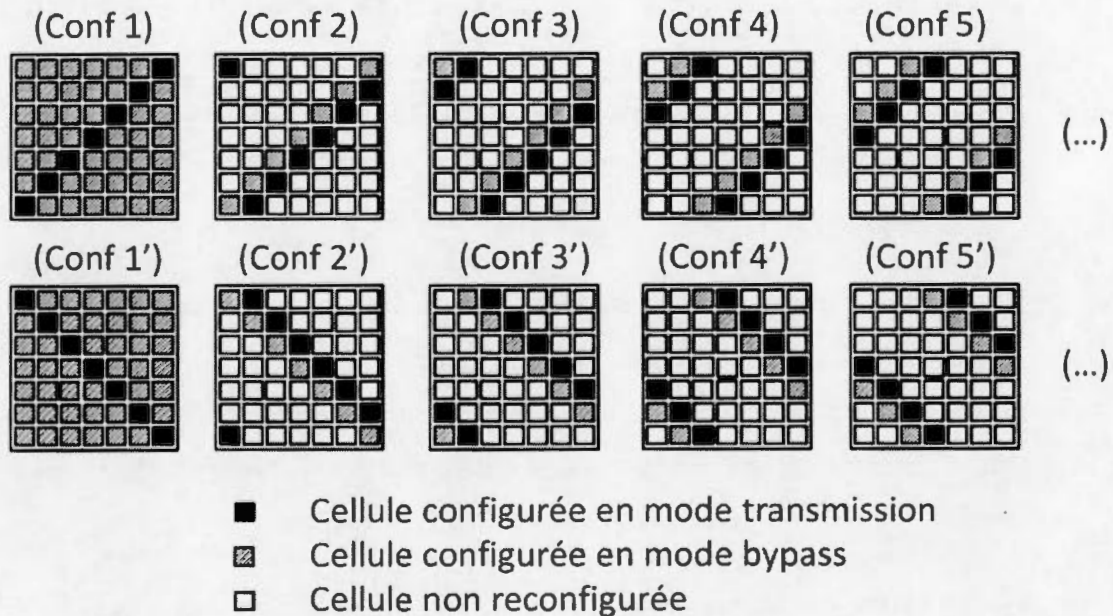


Figure 3.7 **Algorithme de diagnostic en diagonale des liens du FPIN sur un réticule de 7×7 cellules**

L'algorithme de diagnostic en diagonale des liens du FPIN est divisée en trois principales étapes (Fig. 3.8) :

1. **La première étape** configure toutes les cellules en mode écoute, à l'exception de la première diagonale sous test. Le mode écoute permet de récupérer les syndromes de test de tous les liens entrant de la cellule. La diagonale sous test est configurée en mode transmission, afin de transmettre les vecteurs de test sur ses liens d'interconnexions sous test sortants (Fig. 3.7 Conf 1).
2. **La seconde étape** est l'insertion des vecteurs de test par les registres d'insertions TCI de la diagonale sous test.
3. **La troisième étape** est la récupération des syndromes par les registres d'extractions TCO de toutes les cellules en mode écoute afin d'en analyser les résultats.

Le cycle d'insertion des vecteurs de test et d'extraction des syndromes se poursuit jusqu'à avoir testé tous les liens des cellules de la diagonale sous test par l'algorithme du « *Walking one* ». Pour compléter la couverture des pannes de ce diagnostic, le test complémentaire est appliqué dans une étape 2' et 3' (Fig. 3.8) sur la même diagonale avec les vecteurs de l'algorithme « *Walking zero* ».

À la suite de l'application de l'algorithme sur la diagonale sous test, cette dernière est reconfigurée en mode écoute, alors qu'une nouvelle diagonale sous test est reconfigurée en mode transmission (Fig. 3.7 Conf 2). Le test est à nouveau appliqué, et ainsi de suite jusqu'à ce que toutes les diagonales soient testées (Fig. 3.7 Conf 2 à Conf 5).

La couverture des pannes de ce test sur les liens du FPIN comprend tous les collages et pratiquement tous les courts-circuits. Cependant, le court-circuit entre un lien sous test et un lien entrant dans une cellule de la diagonale sous test ne peut être testé, car la cellule n'est pas en mode écoute (Fig. 3.9 (a)). Ce type de court-circuit peut être localisé en testant la cellule en question par une nouvelle diagonale perpendiculaire, assurant ainsi qu'au moins l'une des deux diagonales aura la cellule du lien entrant en mode écoute. Ainsi, lors d'un test complet, chaque cellule est diagnostiquée par deux configurations de diagonales complémentaires afin de permettre un diagnostic complet des liens du FPIN. De même, si deux courts-circuits provenant de deux liens de cellules en mode transmission affectent un même lien, l'application des diagonales complémentaires permettra de les différencier.

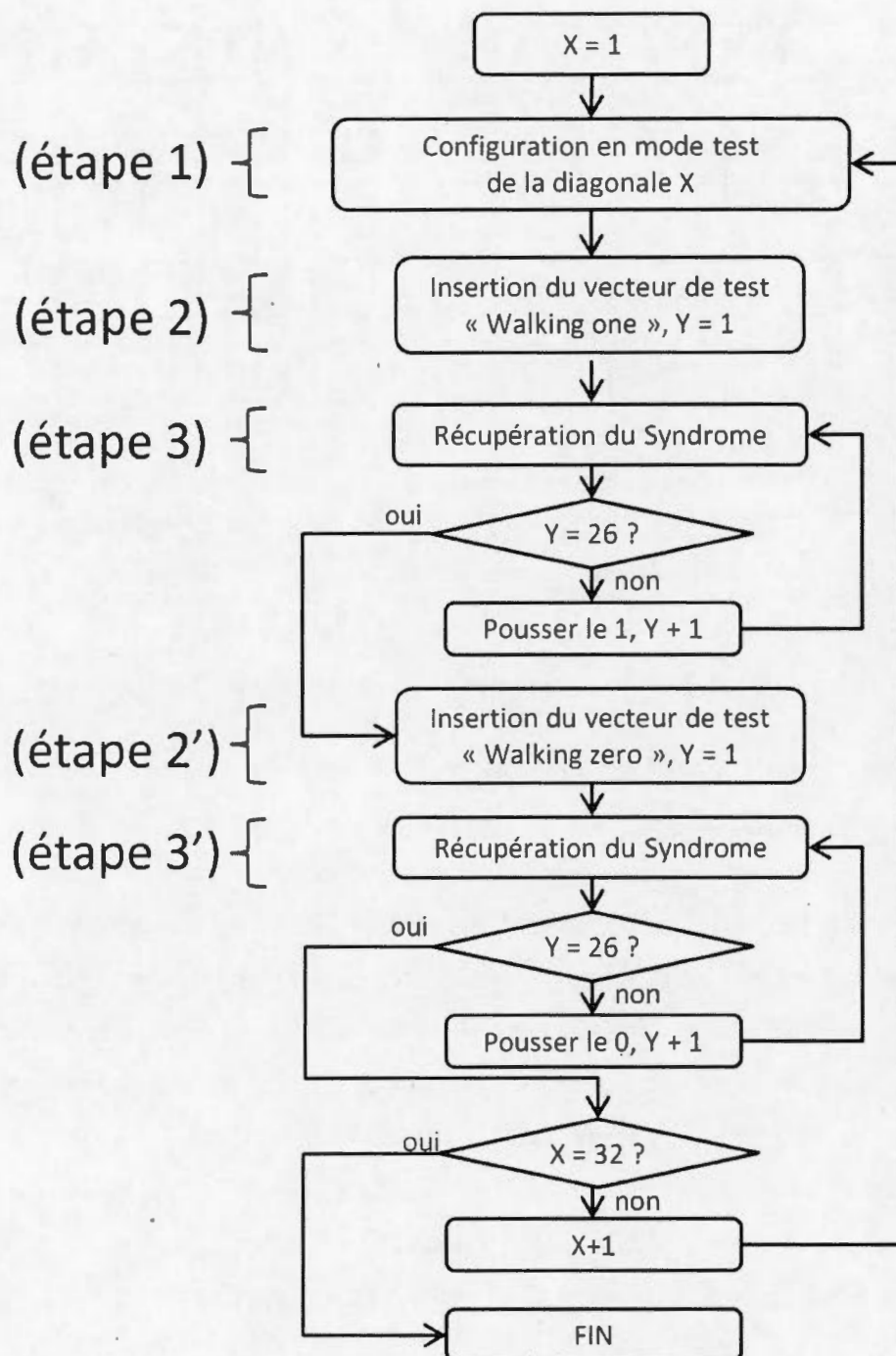


Figure 3.8 Algorithme de diagnostic diagonalisé des liens du FPIN

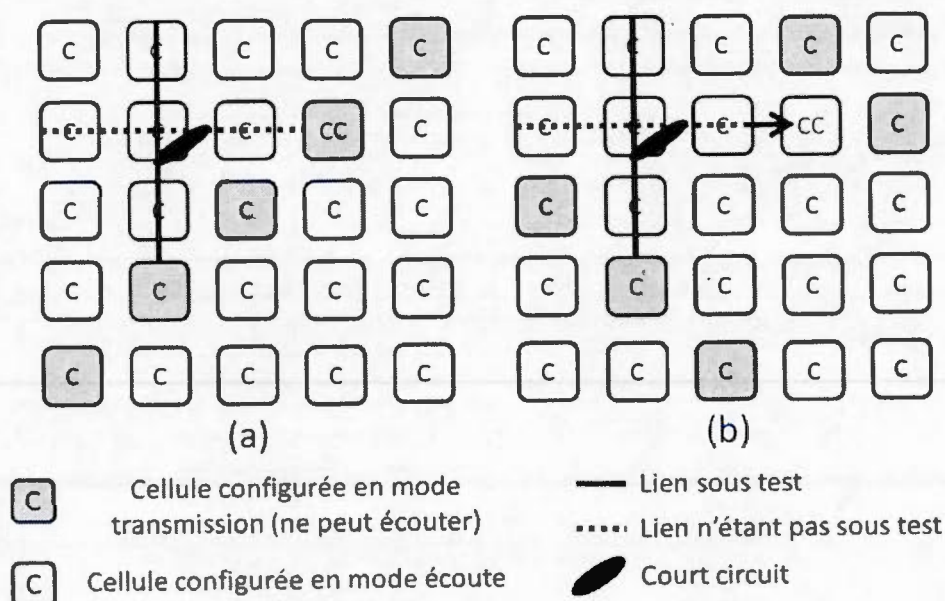


Figure 3.9 Test des configurations complémentaires

L'algorithme de diagnostic des liens proposé dans (Basile-Bellavance, 2009) utilise un vecteur de la taille du registre d'insertion (TCI) composé d'un '1' entouré de '0's afin d'appliquer l'algorithme du « *Walking one* » (l'inverse dans le cas du « *Walking zero* »). Cela implique qu'un seul lien soit testé par cellule lors de chaque test. L'algorithme proposé dans ce mémoire tire avantage de l'architecture du FPIN, et du fait que les liens soient dans quatre directions pour réduire le nombre de tests nécessaire. Ainsi, le vecteur de test est formé de deux blocs identiques, et composé de '1' entouré de '0's. Ce nouveau vecteur à la taille du registre TCI, et permet de tester simultanément un lien horizontal et un lien vertical sans altérer le cône d'influence, ce qui réduit par deux le nombre d'insertions nécessaires pour pousser le '1'. Il est aussi possible de former un vecteur scindé en 4 blocs identiques de '1' entourés de '0's, ce qui permet de tester les quatre directions en même temps (Nord, Sud, Est, et Ouest). Cependant, le test des quatre directions simultanées implique une

superposition des cônes d'influence et une réduction de la couverture des courts-circuits.

Si le temps de diagnostic de l'algorithme proposé ne rencontre pas les contraintes de temps, il est possible de réduire ce temps par l'utilisation d'un test multi-diagonale. Comme les liens sont distants de 2, 4, 8, 16 ou 32 cellules, il est alors possible de séparer chaque diagonale par deux cellules sans rencontrer aucun conflit. Ainsi, le nombre de configurations est réduit à 6 au lieu de 64, et si l'on néglige les courts-circuits non couverts sans la diagonale perpendiculaire, le nombre de configurations est réduit à 3 au lieu de 32. Cependant, la réduction du temps de test réduit la couverture des courts-circuits par la superposition des cônes d'influences.

La section suivante présente l'analyse du temps de diagnostic des *crossbars* et des liens du FPIN, et compare ces temps à ceux de l'algorithme proposé dans (Basile-Bellavance, 2009). Cette section présente aussi l'analyse de complexité de l'algorithme de diagnostic du FPIN.

3.3 Analyse du temps de test et de la complexité de l'algorithme de diagnostic

Le FPIN est composé de plusieurs réticules identiques, interconnectés et répétés dans les directions horizontales et verticales. Chaque réticule dispose de ports JTAG permettant la configuration et le diagnostic du FPIN par la construction de chaînes JTAG. Il est ainsi possible d'insérer et d'extraire des vecteurs de configuration et de test par l'intermédiaire du protocole JTAG sur chaque réticule en parallèle. Ainsi, le temps de diagnostic peut-être défini en nombre de cycles du protocole JTAG nécessaire à insérer ou extraire les valeurs des registres. Pour simplifier l'étude de complexité du temps de test, l'analyse ne prend pas en compte les cycles nécessaires pour passer entre les différents états de la machine à états finis du contrôleur JTAG. Cependant, leur nombre est négligeable, et n'a qu'un impact minime sur cette analyse qui donne des résultats très probants. L'analyse se concentre sur l'application sur un

seul réticule de l'algorithme de diagnostic, car ce diagnostic peut se faire en parallèle sur tous les réticules.

3.3.1 Estimation du temps de diagnostic des commutateurs

L'algorithme de diagnostic des commutateurs programmables, décrit à la section 3.1, nécessite trois étapes, la configuration en mode *broadcast* du commutateur, l'insertion du vecteur de test à l'entrée du commutateur et l'extraction du syndrome de sortie du commutateur. Le temps de test en cycles d'horloge de la machine à états fini JTAG, $T_{\text{commutateurs}}$, de l'algorithme de diagnostic des commutateurs requiert donc le temps de configuration des commutateurs (T_c), le temps d'insertion des vecteurs de test (T_i), et le temps d'extraction des syndromes (T_e) et cela pour une chaîne JTAG passant par les $N \times N$ cellules du réticule de côté N cellules.

$$T_{\text{commutateurs}} = T_c + T_i + T_e \quad (3.1)$$

T_c représente le temps de configuration des commutateurs, et il est défini par :

$$T_c = (N^2 \times CR) \times TCI \quad (3.2)$$

où CR est la taille du registre de configuration du commutateur, N^2 est le nombre de registres à configurer, et TCI le nombre de configurations du commutateur correspondant au nombre d'entrées de celui-ci, soit une configuration en mode *broadcast* pour chaque entrée à tester.

T_i représente le temps d'insertion des vecteurs de test de la taille du registre TCI pour chacune des N^2 cellules et cela pour les deux vecteurs complémentaires et les TCI configurations du commutateur. Soit :

$$T_i = 2(N^2 \times TCI) \times TCI \quad (3.3)$$

En faisant de même pour le temps d'extraction T_e , on obtient :

$$T_e = 2(N^2 \times TCO)TCI \quad (3.4)$$

où N^2 représente le nombre de syndromes de taille TCO à extraire et cela pour les deux vecteurs complémentaires et les TCI configurations du commutateur.

Ainsi, le temps total de diagnostic des commutateurs est :

$$T_{\text{commutateurs}} = (N^2 \times TCI)(CR + 2(TCI + TCO)) \quad (3.5)$$

Selon l'équation 3.5, le temps de configuration des commutateurs (T_c), et le temps de diagnostic ($T_i + T_e$) est dominé par le nombre de cellules qui forme le coté d'un réseau carré, N . Ainsi, le temps de diagnostic d'un réseau augmente de façon quadratique avec ce nombre N .

3.3.2 Estimation du temps de diagnostic des liens du FPIN

L'algorithme de diagnostic des liens (Sec. 3.2) nécessite trois étapes : la configuration de toutes les cellules dans le mode nécessaire au test (écoute ou transmission), l'insertion des vecteurs de test dans les cellules en mode transmission, et l'extraction des syndromes de toutes les cellules en mode écoute. Le temps de test en cycles d'horloge de la machine à état fini JTAG, T_{lien} , de l'algorithme de diagnostic des liens requiert donc le temps de configuration des commutateurs (T_c), le temps d'insertion des vecteurs de test (T_i), et le temps d'extraction des syndromes (T_e) et cela appliqué aux $N \times N$ cellules du réseau de coté N cellules.

$$T_{\text{lien}} = T_c + T_i + T_e \quad (3.6)$$

T_c représente le temps de configuration des cellules afin d'appliquer les vecteurs de test. Cette première étape demande une première configuration de toutes les cellules afin de toutes les configurer en mode écoute à l'exception de la diagonale sous test qui est configuré en mode transmission. Puis, les configurations suivantes ne

demandent que la configuration de l'ancienne diagonale en mode écoute et de la nouvelle diagonale en mode transmission. Ainsi, T_c est défini par :

$$T_c = (N^2 \times CR) + (2N-1)(2N)CR \quad (3.7)$$

où $(N^2 \times CR)$ représente la configuration initiale de toutes les cellules, $(2N)CR$ représente la taille du registre de configuration d'une nouvelle diagonale, et $(2N-1)$ est le nombre de fois que l'on déplace la diagonale et sa complémentaire pour effectuer le test.

T_i représente le temps total d'insertion des vecteurs de test de l'algorithme « *walking one* », soit :

$$T_i = 2N(2(N \times TCI + SHIFT)) \quad (3.8)$$

avec

$$SHIFT = ((TCI-2)s^{-1})-1 \quad (3.9)$$

Où $(N \times TCI)$ représente l'insertion du premier vecteur de test dans le registre TCI de la diagonale. Ensuite, par l'application de l'algorithme « *Walking one* », le vecteur de test est décalé SHIFT fois en fonction du nombre de tests simultanés s . Enfin, le test est appliqué une seconde fois pour son complément, l'algorithme « *Walking zero* ». Le tout est répété pour les $2N$ configurations de diagonale. Le nombre de décalages du vecteur de test SHIFT est défini par le nombre de liens à tester, soit $TCI-2$ car on exclut les liens internes de la cellule. Ce vecteur doit être décalé en fonction du nombre de '1' qu'il comporte, soit $((TCI-2)s^{-1})-1$, avec s le nombre de '1' dans le vecteur de test, et auquel on extrait le premier test dont le vecteur initial est à l'origine.

T_e , le temps total d'extraction des symboles est défini par :

$$T_e = 2N((N^2 - N)TCO)2(SHIFT+1) \quad (3.10)$$

où $(N^2-N)TCO$ représente tous les registres TCO à extraire moins la diagonale qui génère les vecteurs de test, $2(SHIFT+1)$ est le nombre de tests effectués, ce qui est déduit de (3.8), et le tout pour les $2N$ tests.

On constate des équation 3.7 et 3.8, que le temps de configuration et d'insertion des vecteurs de test sont toujours dominés par le facteur N^2 . Cependant, le facteur dominant de l'équation 3.6 du temps de diagnostic des liens du FPIN provient de l'équation 3.10, avec un facteur cubique N^3 . Ce facteur bien que dominant a été réduit par l'utilisation du diagnostic diagonalisé, alors qu'il était de N^4 dans l'algorithme proposé par de (Basile-Bellavance, 2009).

3.3.3 Analyse de complexité

Le tableau 3.2 résume et compare les temps de diagnostic des algorithmes de (Basile-Bellavance, 2009) (nommé « Basile ») et ceux proposés dans le présent mémoire. L'analyse de ces temps et complexités se base sur le diagnostic d'un réseau traversé par une chaîne JTAG passant par toutes ces cellules. L'unité utilisée est le nombre de cycles d'horloge (*clock cycle*, cc) de la machine à états finis du contrôleur JTAG. Pour simplifier l'étude, l'analyse ne prend pas en compte les cycles nécessaires pour passer entre les différents états de la machine à états finis du contrôleur JTAG, mais seulement les cycles d'horloge nécessaires à l'insertion et à l'extraction des vecteurs de configuration et de test. Cependant, leur nombre est négligeable et n'a qu'un impact minime sur cette analyse.

Dans l'algorithme proposé dans ce mémoire, la complexité temporelle de l'algorithme de diagnostic des commutateurs est de $O(N^2)$ pour la configuration (T_c) (Eq. 3.2), et de $O(N^2)$ pour le test (T_i+T_e) (Eq. 3.3 et Eq. 3.4), ce qui ne diffère pas de l'algorithme précédent. La nouvelle architecture qui dispose de registre CR, TCI et TCO plus petit que l'architecture précédente permet toutefois une réduction du temps de diagnostic des commutateurs de près de 1.8 fois. La complexité temporelle de l'algorithme proposé dans (Basile-Bellavance, 2009) est dominée par l'extraction des

syndromes du test des liens ($O(N^4)$), alors que l'algorithme de diagnostic des liens du FPIN proposé dans ce mémoire est aussi dominé par l'extraction des syndromes, l'algorithme diagonalisé permet une complexité de $O(N^3)$. Ceci permet une réduction du temps de diagnostic des liens de près de 130 fois. Finalement, notre algorithme permet ainsi une réduction de 112 fois du temps de diagnostic des commutateurs et des liens du FPIN pour le test complet d'un réseau de 32×32 cellules.

Tableau 3.2 Résumé des temps de test et de la complexités des l'algorithmes de diagnostic pour un réseau de 32×32 cellules

| | | Basile A | Proposé B | Facteur de réduction (A/B) |
|--------------|--|-------------|--------------|----------------------------------|
| commutateurs | T_c | 8.4 M cc | 3.8 M cc | 2.2 |
| | | $O(N^2)$ | $O(N^2)$ | |
| | T_i | 3.3 M cc | 1.3 M cc | 1.3 |
| | | $O(N^2)$ | $O(N^2)$ | |
| | T_e | 0.1 M cc | 1.4 M cc | |
| | | $O(N^2)$ | $O(N^2)$ | |
| | $T_{\text{commutateurs}}$ | 11.8 M cc | 6.5 M cc | 1.8 |
| liens | T_c | 1.32 M cc | 0.7 M cc | 1.9 |
| | | $O(N^2)$ | $O(N^2)$ | |
| | T_i | 3.4 M cc | 0.1 M cc | 34 |
| | | $O(N^2)$ | $O(N^2)$ | |
| | T_e | 5.6 G cc | 42.6 M cc | 131 |
| | | $O(N^4)$ | $O(N^3)$ | |
| | T_{liens} | 5.6 G cc | 43.4 M cc | 129 |
| | $T_{\text{commutateurs}} + T_{\text{liens}}$ | 5,61 G cc | 49.9 M cc | 112.4 |

3.4 Résumé

Ce chapitre a détaillé l'application de l'algorithme de diagnostic des commutateurs et des liens du FPIN proposé dans ce mémoire.

L'algorithme de diagnostic des commutateurs programmable proposé ici est en principe le même que celui proposé par (Basile-Bellavance, 2009), et seule l'application le différencie de par l'architecture différente des registres de configuration, d'insertion et d'extraction des commutateurs. Il n'y a donc aucun gain de temps significatif sur l'insertion des vecteurs de test et l'extraction des syndromes, car ces registres sont de tailles comparables. Cependant, la programmation des commutateurs qui demande 314 bits par commutateur dans l'architecture précédente, n'en demande plus que 142 bits dans l'architecture utilisée dans ce mémoire, soit une réduction de 55 % du temps de configuration des commutateurs programmables.

L'algorithme de diagnostic des liens du FPIN proposé dans ce mémoire s'appuie sur l'architecture du FPIN pour redéfinir le cône d'influence des tests, et en optimiser l'utilisation par un algorithme diagonalisé. Ainsi, l'algorithme permet de paralléliser N tests des liens pour un réseau de $N \times N$ cellules alors qu'un seul test par réseau n'est possible dans l'algorithme précédant. Ceci permet de réduire la complexité à ($O(N^3)$) par rapport à l'algorithme proposé par (Basile-Bellavance, 2009) dont la complexité du diagnostic des liens est de ($O(N^4)$). Pour permettre une couverture des pannes complète des liens, le test doit être complété par les diagonales perpendiculaires. Ce doublement des tests est cependant compensé par l'optimisation faite de l'application de l'algorithme « *Walking one* », où deux liens perpendiculaires sont testés en même temps sans réduire la couverture des pannes. Ceci permet une réduction du temps de diagnostic des liens de près de 130 fois pour un test complet de réseau de 32×32 cellules.

Finalement, notre algorithme permet ainsi une réduction de 112 fois du temps de diagnostic des commutateurs et des liens du FPIN pour le test complet d'un réseau de 32×32 cellules.

CHAPITRE IV

RÉSULTATS

Le projet de recherche DreamWafer™ a déjà permis de mettre à jour plusieurs versions du WaferIC™. Le prototype de la version 3 développé auparavant par l'équipe DreamWafer, et appelé MiniWaferIC, est utilisé dans ce mémoire sous la forme d'un circuit intégré de deux par deux réticules connectés à un PowerBlock. Une équipe composée de Safa Berrima, Sylvain Charasse et Gontran Sion a été formée afin de tester ce prototype. L'objectif était de valider le fonctionnement du WaferIC et du PowerBlock. Le fait de rendre le prototype fonctionnel a aussi permis de valider l'algorithme de diagnostic proposé dans le chapitre 3. Ainsi, un environnement de test a été mis en place à l'aide de Sylvain Charasse. Cet environnement de test permet de contrôler le circuit sous test MiniWaferIC.

Ce chapitre présentera dans une première section l'implémentation physique de l'environnement de test et la vérification de fonctionnement primaire du MiniWaferIC. Dans une deuxième section sera présenté l'outil logiciel conçu pour configurer et valider le WaferIC en générant les vecteurs de test et récupérant les syndromes issus de ces tests. L'outil logiciel permet aussi d'automatiser les tests et de générer des rapports d'étape afin de vérifier le bon fonctionnement des tests. Une troisième section présentera les résultats des tests effectués, ainsi que leur analyse. Le chapitre se terminera par une conclusion.

4.1 Description de l'environnement de test

4.1.1 Connexions d'alimentation et de communication au MiniWaferIC

Le MiniWaferIC est un système composé d'une version réduite du VLAIC WaferIC. Cette version réduite a été composée pour s'adapter à un PowerBlock qui ne permet d'alimenter que 2 groupes de 2 réticules, et est donc formé d'un circuit intégré de deux par deux réticules. Outre l'alimentation, le PowerBlock contrôle la communication avec les réticules au travers de chacun de leur bus JTAG, un par réticule. Cette communication est gérée par le FPGA du PowerBlock (un Igloo AGL060V5 de Microsemi, anciennement Actel). Les réticules sont alimentés deux à deux par deux alimentations distinctes. Un rail d'alimentation 1.8 V provenant directement de l'environnement de test, et un rail d'alimentation 3.3 V provenant du convertisseur DC/DC (IP1202) du PowerBlock et alimenté en 12 V de l'environnement de test. Bien que le convertisseur DC/DC permette des tensions variables sur ses deux sorties, il est convenu dans ce mémoire que ces tensions seront toujours maintenues à 3.3 V. Ce convertisseur DC/DC est contrôlé par le FPGA. Il est à noter que le rail d'alimentation directe 1.8 V des réticules ne doit être actif que si le rail d'alimentation 3.3 V du convertisseur est supérieur à 1.8 V, car il y a des jonctions *pn* entre les deux rails d'alimentations qui se doivent d'être alimentées en inverse afin d'éviter tout problème de *latchup* dans le WaferIC. En plus des rails d'alimentations des quatre réticules (3.3 V et 1.8 V), deux autres rails d'alimentations sont nécessaires au FPGA. Ainsi, le cœur du FPGA est alimenté en 1.5 V, alors que ses broches d'entrées et sorties le sont en 3.3 V. Ces deux alimentations indépendantes proviennent de l'environnement de test. Le module WaferIC/PowerBlock est présenté à la figure 4.1, l'environnement de test est présenté à la figure 4.2 et les alimentations utilisées sont présentées à la figure 4.3.

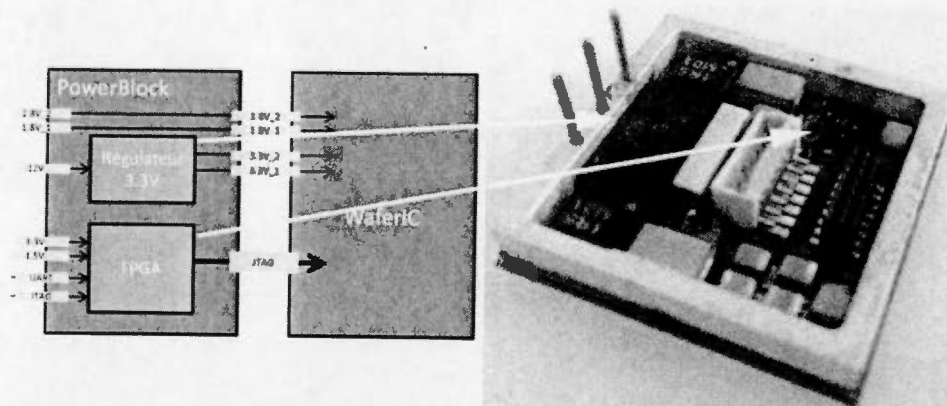


Figure 4.1 Module WaferIC/PowerBlock, appelé MiniWaferIC

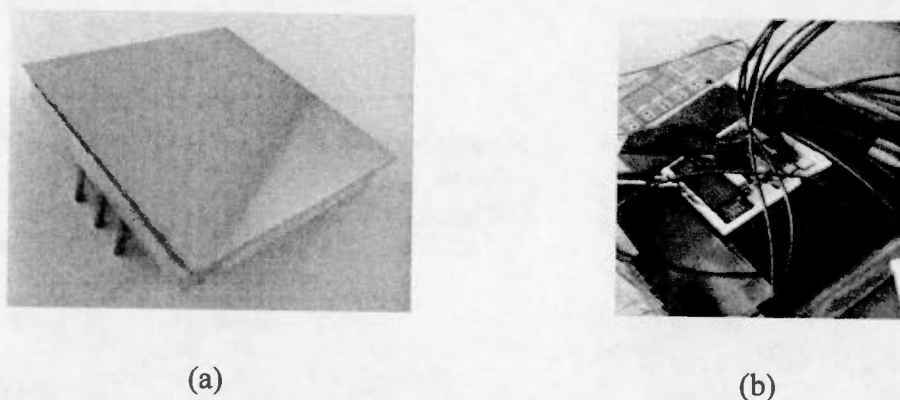


Figure 4.2 Module MiniWaferIC (a) et son environnement de test (b)



Figure 4.3 Alimentations de l'environnement de test

4.1.2 Test de fonctionnement du MiniWaferIC

Les tests présentés ici sont synthétisés afin de permettre une compréhension des travaux réalisés. Cependant, les tests étaient réalisés très progressivement et de façon méticuleuse sur le seul prototype existant pour ce projet. Un second prototype a existé, mais a été détérioré dès le début des tests. Le rapport technique « Test du MiniWaferIC » rapporte de façon plus exhaustive les étapes de ces tests. Le schéma du système est représenté à la figure 4.4. Une liste complète des connexions du MiniWaferIC est faite à la figure 4.5, alors qu'une vue complète de l'environnement de test est présentée à la figure 4.6.

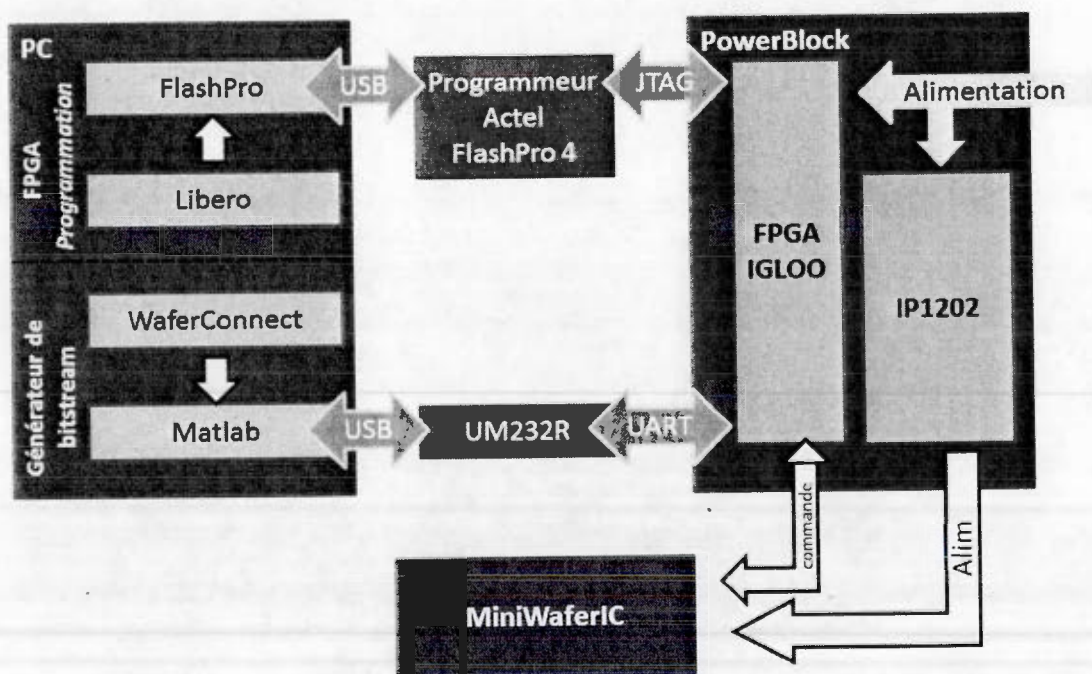


Figure 4.4 Schématique de l'environnement de test du MiniWaferIC

| Connecteur CON4 | Signal | Module | Nom schématique |
|--------------------|-----------|-------------------------|------------------|
| 1 | GND | GND | GND |
| 2 | CLK0 | UM232R | BOTTOM_PCB_CLK0 |
| 3 | CLK | Actel FlashPro4 | AGL060-TCK |
| 4 | CLK1 (NC) | UM232R | BOTTOM_PCB_CLK1 |
| 5 | GND | GND | GND |
| 6 | DOUT0 | UM232R | BOTTOM_PCB_DOUT0 |
| 7 | TDO | Actel FlashPro4 | AGL060-TDO |
| 8 | DOUT1 | UM232R | BOTTOM_PCB_DOUT1 |
| 9 | TDI | Actel FlashPro4 | AGL060-TDI |
| 10 | DIN0 | UM232R | BOTTOM_PCB_DIN0 |
| 11 | TMS | Actel FlashPro4 | AGL060-TMS |
| 12 | DIN1 | UM232R | BOTTOM_PCB_DIN1 |
| 13 | GND | GND | GND |
| 14 | TRST | Actel FlashPro4 | AGL060-TRST |
| 15 | 3.3V | Alimentation externe | +3.3V |
| 16 | 1.5V | Alimentation externe | +1.5V |

Figure 4.5 Connexions du PowerBlock

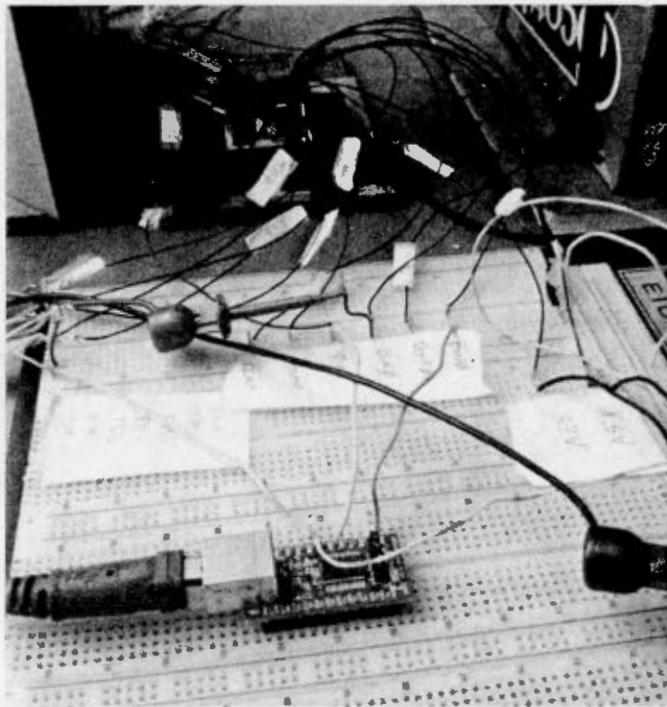


Figure 4.6 Environnement de test du MiniWaferIC

Le test de fonctionnement du MiniwaferIC se divise en trois étapes :

1. La validation du fonctionnement du FPGA du PowerBlock responsable de trois tâches, la communication avec l'ordinateur, la gestion du protocole JTAG des réticules et la gestion des deux rails d'alimentation 3.3 V des réticules provenant du 12V de l'environnement de test.
2. La validation de la bonne alimentation du MiniWaferIC en s'assurant que les courants consommés se trouvent dans des intervalles raisonnables.
3. La validation du fonctionnement logique du MiniWaferIC en réalisant une première configuration simple.

Le dernier test sera présenté dans la section 4.2, car il est lié au développement de l'outil logiciel.

4.1.2.1 Test et programmation du FPGA

Le premier test de fonctionnement du FPGA a été de faire lire ses registres de configurations par le programme Libero de Microsemi. Cette connexion a été établie au travers du module FlashPro4 de Microsemi alors que seul le FPGA du PowerBlock était alimenté. Ce test a permis de confirmer le bon fonctionnement du FPGA.

Le deuxième test a été réalisé en établissant une connexion entre l'outil logiciel développé sous Matlab® et le FPGA par l'intermédiaire du convertisseur USB/UART UM232R (FTDI, 2011). Le FPGA n'avait alors été programmé que pour connecter son entrée RX à sa sortie TX afin de renvoyer le paquet directement à l'ordinateur. Ce test permettait de vérifier, en plus du fonctionnement du FPGA, le bon fonctionnement du lien de communication long d'une cinquantaine de centimètres.

Enfin, un bloc de communication RX/TX de type UART connecté à une machine à états finis a été implémenté dans le FPGA afin de pouvoir traiter les paquets reçus de l'outil logiciel. Le bloc de communication RX/TX utilise un signal d'horloge provenant d'un générateur de fonction externe afin de se synchroniser. Une fois le bloc de communication fonctionnel, un bloc de contrôle du convertisseur DC/DC a été implémenté dans le FPGA afin de contrôler les deux tensions de sorties 3.3 V du contrôleur DC/DC, de même que de pouvoir lire sa broche PGOOD afin de s'assurer du bon fonctionnement du contrôleur DC/DC. Un dysfonctionnement de la broche PGOOD du convertisseur DC/DC n'a pas permis d'implémenter cette fonctionnalité. Suite aux tests effectués, et à la sensibilité du convertisseur DC/DC, seule la configuration des deux sorties en 3.3 V a été gardée dans l'outil logiciel, car le changement de tension en sortie faisait arrêter inopinément le convertisseur DC/DC. Cet arrêt de fonctionnement pouvait effectivement entraîner la surchauffe et un endommagement des réticules.

4.1.2.2 Alimentation du MiniWaferIC

Le test de l'alimentation du MiniWaferIC s'est résumé à appliquer les tensions sur ses deux rails d'alimentations et d'en vérifier les courants consommés. Dans un premier temps l'alimentation en 3.3 V provenant du convertisseur DC/DC a été appliquée sous le contrôle du FPGA, puis l'alimentation 1.8 V a été appliquée directement au travers de l'environnement de test. Les courants observés sur les rails d'alimentation des réticules étaient de 0,7 A pour l'alimentation 12 V et de 3.96 A pour l'alimentation 1.8 V. La remise à zéros des quatre réticules par le maintien du signal TRST de la connexion JTAG à '0' a permis une réduction du courant consommé par l'alimentation 12 V de l'environnement de test à 0.42 A. Cette réduction pouvant provenir de la mauvaise configuration à la mise sous tension des réticules entraînant des courts-circuits au niveau des NanoPads. La puissance totale consommée par le MiniWaferIC à l'état remis à zéro représente donc à peu près 12W, ce qui a été jugé raisonnable.

4.1.2.3 Caractéristique thermique du MiniWaferIC

La puissance de 12W consommée par le MiniWaferIC, associée à la fois à sa taille et à son faible coefficient de dissipation thermique, implique une augmentation rapide de sa température. Une température élevée des circuits intégrés contribuant à leurs vieillissements prématurés, un système de refroidissement a été mis en place pour assurer une durée de vie optimale au système. Ainsi, l'installation d'un ventilateur, présenté à la figure 4.7, a permis une réduction de 11 °C du MiniWaferIC en fonctionnement statique, le faisant passer de 67 °C à 56 °C. Cette réduction de température sera un atout pour les tests de longue durée et permettra certainement une augmentation de la durée de vie du module.

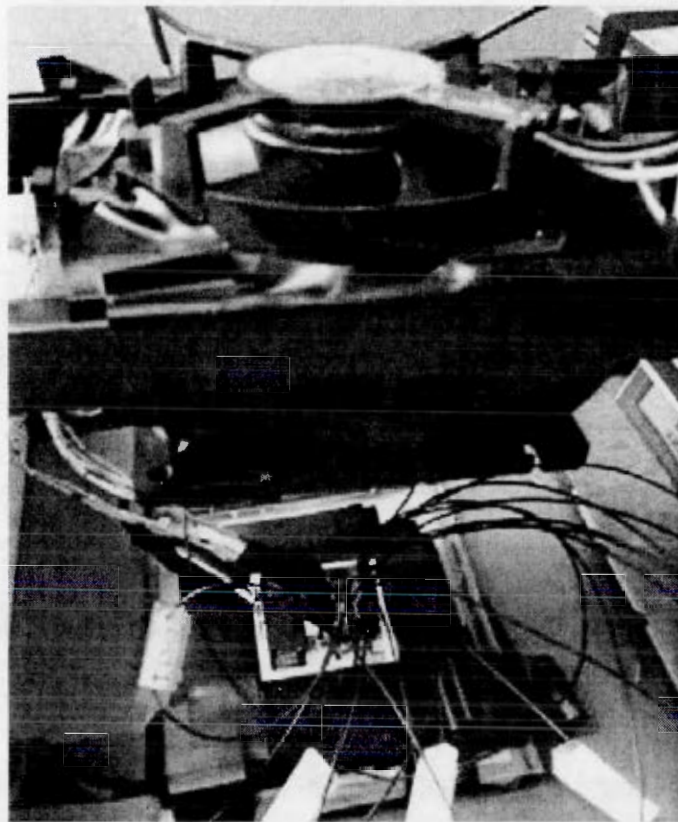


Figure 4.7 **Système de refroidissement du MiniWaferIC**

4.2 Outil logiciel pour l'analyse et l'automatisation

Afin de gérer les différentes étapes nécessaires à la configuration des réticules du MiniWaferIC en vue du diagnostic de son FPIN, un outil logiciel sous l'environnement Matlab® a été développé. En plus de générer et transmettre les chaînes JTAG nécessaires à la configuration des réticules pour effectuer le test et diagnostic, le programme génère des fichiers d'étapes lors du processus et enregistre tous les signaux JTAG dont la sortie TCO. Cette section présente une vue générale de cet outil logiciel, puis détaille ses fonctions.

4.2.1 Présentation générale de l'outil logiciel

Le but de l'outil logiciel développé est de permettre :

- la communication avec le FPGA du PowerBlock;
- le contrôle du convertisseur DC/DC du PowerBlock afin d'alimenter les réticules en 3.3 V;
- la génération des chaînes JTAG afin de configurer et tester les réticules;
- la récupération des syndromes pour une analyse de diagnostic ultérieure.

La communication de l'outil logiciel avec le FPGA est faite par le port USB de l'ordinateur qui est par la suite converti au protocole UART par le module UM232R. L'outil logiciel prend en entrée un script de test permettant la génération des chaînes JTAG nécessaire au FPGA pour contrôler le réticule sous test. L'utilisation du programme Realterm, un terminal disposant de fonctions avancées, a été choisie pour la transmission USB des fichiers de chaînes JTAG et la récupération, dans un nouveau fichier, des signaux TCO en plus des signaux déjà envoyés (TCK, TMS, TRST et TCI). Ces fichiers sont générés par chaque fonction de l'environnement de test afin de pouvoir en analyser le bon fonctionnement. En plus des fichiers d'étapes de chaque fonction, un fichier réunissant toutes les commandes et un autre concaténant tous les signaux sont générés (GenCom.txt et SignauxInFull.txt

respectivement). Le fichier SignauxInFull.txt contient tous les syndromes des tests, et est le fichier analysé pour le diagnostic du MiniWaferIC. Le flot général des tests est présenté à la figure 4.8.

4.2.2 Fonction de remise à zéro

Le fichier d'entrée d'un test commence par une fonction de remise à zéro du réticule afin d'être dans un état de réticule connu. La remise à zéro permet d'être dans un état où tous les modules sont mis dans leur état initial donc fonctionnellement arrêtés, comme présenté à la section 4.1.2.1 par la baisse de la consommation de courant. Cette fonction ne fait que maintenir le signal TRST à '0' pendant 6 cycles d'horloge de la machine à état fini du contrôleur JTAG.

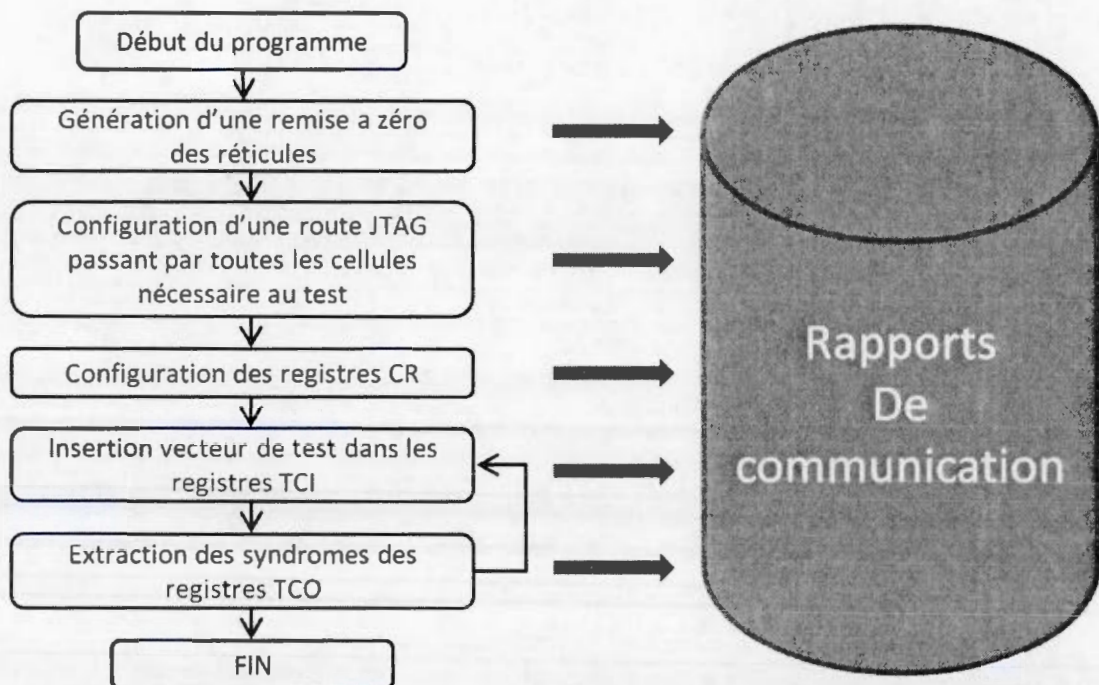


Figure 4.8 Flot générique de l'outil logiciel pour le diagnostic du FPIN

4.2.3 Fonction de construction de la chaîne JTAG

Une des étapes fondamentales dans le contrôle des réticules est la construction de la chaîne JTAG passant par les cellules que l'on souhaite configurer (Sec. 2.4). Alors que cette chaîne est statique dans la très grande majorité des circuits intégrés, et donne un accès direct à tous les registres de test et de configuration de ces circuits, la chaîne JTAG est programmable dans le cas du WaferIC, et demande d'être configurée au moins une première fois par les registres Dest (destination) des contrôleurs JTAG (TAP) des cellules, afin d'accéder aux registres du circuit.

La fonction implémentée dans cet environnement de test pour la construction de la chaîne JTAG permet de définir cette chaîne par des nombres représentant ses directions (Est=1, Nord=2, Ouest=3 et Sud=4). Ainsi, la chaîne [1 1 2 2 3 3 4 3] configure la chaîne de la figure 4.9. Une fois la chaîne configurée, tous les registres des cellules que la chaîne JTAG traverse sont accessibles par l'outil logiciel. La construction de la chaîne JTAG démarre de la seule cellule d'un réticule disposant d'une entrée TDI JTAG et se termine par la seule cellule d'un réticule disposant d'une sortie TDO JTAG (aux positions (1,1) et (1,2) respectivement). Ainsi, la plus petite chaîne possible est [2 3], configurant la cellule (1,1) vers le Nord et la cellule (1,2) vers l'Ouest. La configuration de la chaîne JTAG se fait en ajoutant une cellule à la fois à la chaîne, ainsi, pour programmer la chaîne [1 2 3 3], il faut une série de quatre vecteurs de données. Un premier vecteur configure la première cellule (1,1) pour faire pointer la chaîne JTAG vers l'Est, donnant accès à la cellule (2,1). Puis, un deuxième vecteur permet de programmer la nouvelle cellule vers le Nord afin que la chaîne JTAG ait accès à la cellule (2,2), et la première cellule vers l'Est comme auparavant. Et ainsi de suite. La figure 4.10 résume la configuration de la chaîne JTAG [1 2 3 3].

Cette seule fonction a permis l'application et l'automatisation des tests du diagnostic de la chaîne JTAG présenté par (Berrima, 2015).

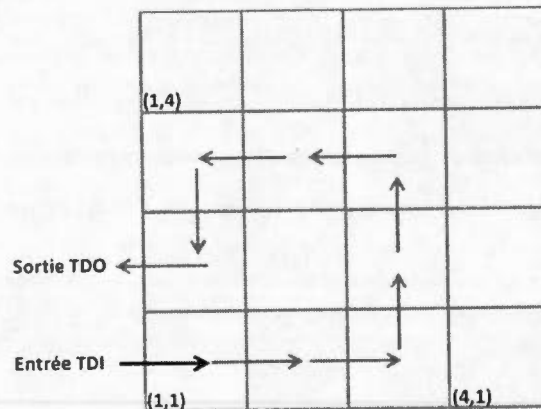


Figure 4.9 Exemple de chaîne configurée par la commande [1 1 2 2 3 3 4 3]

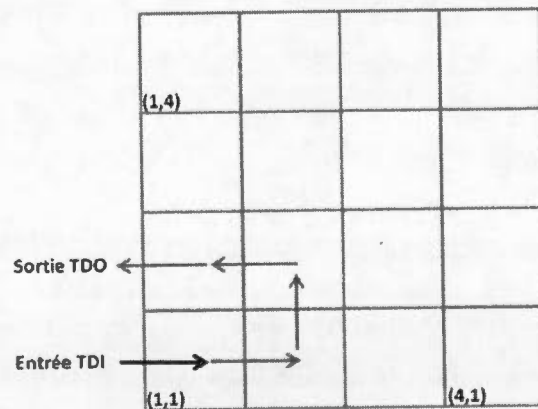


Figure 4.10 Vecteurs de configuration des registres Dest pour la chaîne JTAG [1 2 3 3]

Tableau 4.1 Commandes d'intérêt des modes de registres

| Nom | Mode | Commande |
|-----------------|--------|----------|
| Destination | Dst | 2 |
| Crossbar | CR | 5 |
| Vecteur de test | TCI | 6 |
| Syndrome | TCO | 30 |
| Bypass | Bypass | 31 |

4.2.4 Fonction de configuration des autres registres

L'application des tests du diagnostic proposé demande l'utilisation de trois types de registres particuliers, les registres CR permettant la configuration des *crossbars*, les registres TCI permettant l'insertion des vecteurs de tests, et les registres TCO permettant l'extraction des syndromes. Ces trois types de registres sont accessibles par deux fonctions de l'environnement de test, la fonction de sélection des registres et la fonction de transfert des données. Ces deux fonctions sont aussi utilisées dans la construction des chaînes JTAG, qui n'est qu'un cas particulier de configuration de registres. La fonction de sélection des registres permet de sélectionner l'accès à un type de registre de la cellule (CR, TCI et TCO) (Tab. 4.1). Puis les valeurs de ces registres peuvent être poussées dans l'ordre de la cellule la plus lointaine à la cellule la plus proche dans la chaîne JTAG.

4.3 Résultats et analyses

L'algorithme de diagnostic des commutateurs et des liens du FPIN présenté dans ce mémoire a été testé sur la structure matérielle MiniWaferIC. Ce test s'est divisé en trois étapes distinctes :

1. La construction de chaîne JTAG afin de permettre l'accès à toutes les cellules du réseau.

2. L'application du test des commutateurs.
3. L'application du test des liens.

Ces trois tests physiques sont présentés dans les trois sous-sections suivantes.

4.3.1 Construction de chaînes JTAG

L'une des fonctions les plus importantes de l'outil logiciel est la construction de la chaîne JTAG, car c'est par elle qu'est faite toute la configuration du MiniWaferIC. Cette section en présente les résultats de l'application décrite dans les sections 2.4 et 4.2.3.

La figure 4.11 présente les signaux générés par l'outil logiciel et envoyés pour la construction de la chaîne JTAG [1 2 3 3]. On constate la mise en mode 'Dest' de la première cellule (Fig. 4.11 IR (1)), puis de l'envoi de la valeur de la direction (Fig. 4.11 DR (1)). Pour mettre la seconde cellule en mode 'Dest', on pousse l'instruction 'Dest' dans la première cellule (Fig. 4.11 IR (2)), qui contient déjà cette instruction et va donc être décalée dans la cellule suivante, ainsi les deux cellules de la chaîne JTAG sont en mode 'Dest'. Puis les données de direction des deux cellules à configurer (DR (2)) sont décalées. Et ainsi de suite. Tant que la chaîne JTAG est incomplète, aucun signal TDO ne peut sortir du réseau.

Une fois la configuration de la chaîne JTAG complétée, une deuxième insertion de la dernière commande de configuration de la chaîne JTAG permet d'observer les valeurs des registres sur la sortie TDO (signature). Le fait d'observer des signaux de sortie de la chaîne JTAG permet de confirmer la construction de la chaîne de la cellule d'entrée à la cellule de sortie du contrôleur JTAG du réseau. De plus, la comparaison des valeurs de sorties avec les valeurs de la dernière commande de configuration de la chaîne JTAG permet de vérifier le bon comportement des registres de la chaîne JTAG. La figure 4.12 présente l'observation de signaux (signature) sur la sortie TDO confirmant la bonne construction de chaîne JTAG.

Il est à noter que l'outil logiciel a généré 2 115 584 cycles d'horloge de la machine à états finis du contrôleur JTAG pour permettre la construction de la chaîne JTAG passant par toutes les cellules d'un réseau de 32×32 cellules.

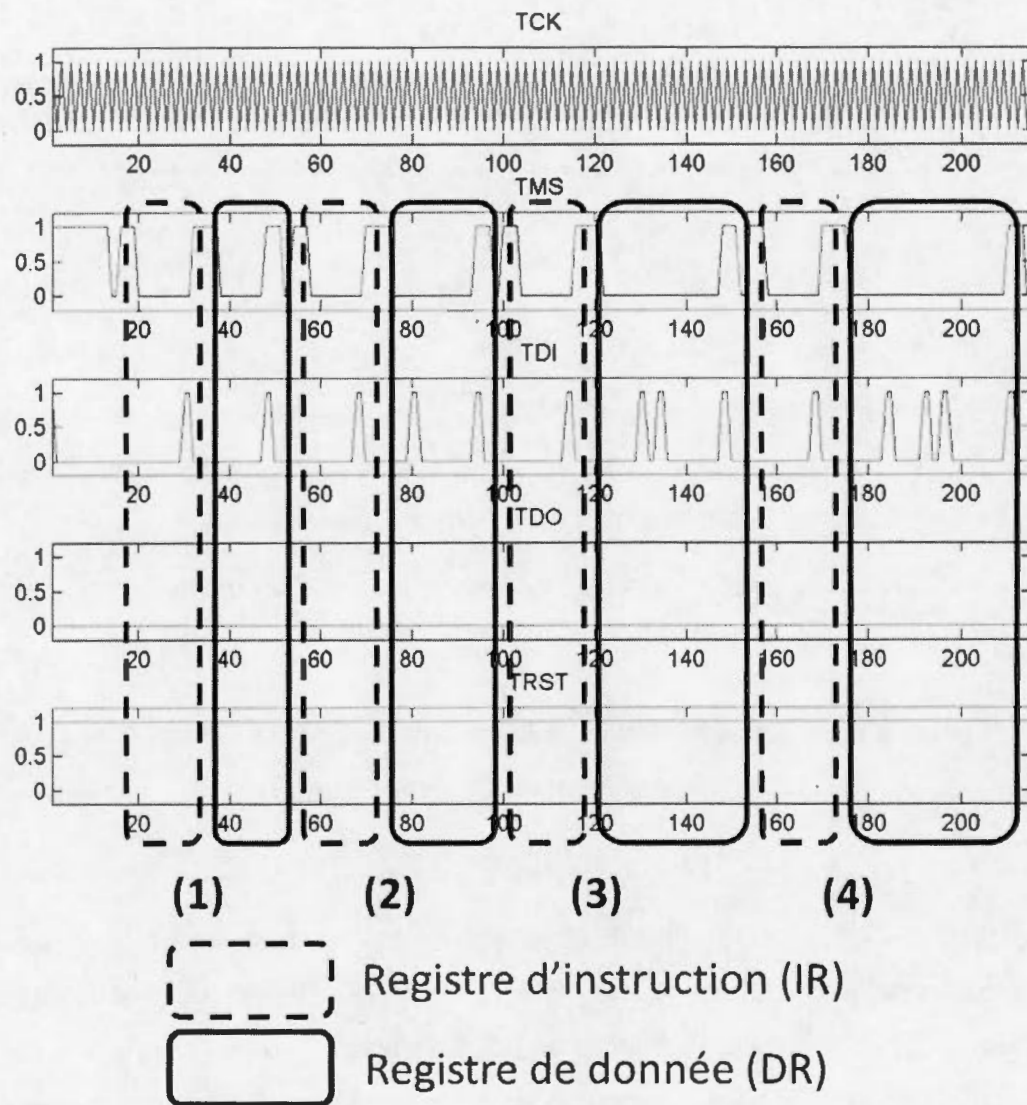


Figure 4.11 Signaux JTAG généré pour la construction de la route [1 2 3 3]

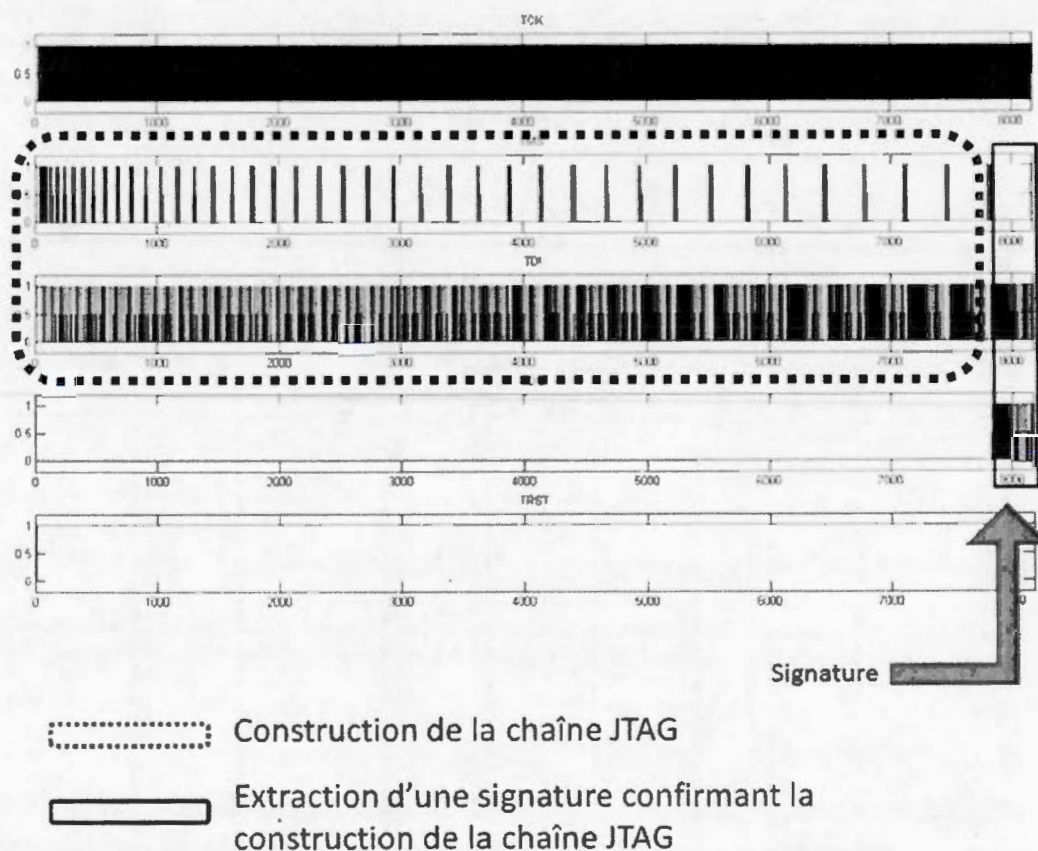


Figure 4.12 Confirmation de la construction d'une chaîne JTAG complète par l'extraction de la signature

4.3.2 Test de l'algorithme de diagnostic des commutateurs (*crossbars*)

L'application de l'algorithme de diagnostic des commutateurs (*crossbars*) demande la configuration de trois types de registres : le registre de configuration des commutateurs, le registre d'insertion des vecteurs de tests et le registre d'extraction des syndromes. Ces configurations de registres se passent en trois étapes comme décrites à la section 3.1 ;

1. la configuration des commutateurs en mode *broadcast*.
2. l'insertion des vecteurs de tests.

3. l'extraction des syndromes.

La figure 4.13 représente une portion du test de l'algorithme de diagnostic des commutateurs appliqué à une chaîne de 4 cellules. On retrouve en premier lieu la configuration des commutateurs en mode *broadcast*, puis 4 blocs représentant deux séries d'insertions de vecteurs de test et d'extractions de syndromes, pour les deux tests complémentaires.

Le test présenté ici a été effectué sur 4 cellules seulement afin de rendre compréhensifs les signaux du contrôleur JTAG de la figure 4.13. L'application du test à un réticule de 32×32 cellules demande à l'outil logiciel de générer 7 376 278 cycles d'horloge de la machine à états finis du contrôleur JTAG. Les calculs théoriques prévoyaient une application en 6,5 millions de cycles. Ainsi, comme que prévu, l'approximation du calcul théorique négligeant le déplacement entre les états de la machine à états finis du contrôleur JTAG a subi une augmentation de 13,5%.

4.3.3 Test de l'algorithme de diagnostic de liens

Les tests du diagnostic des commutateurs (crossbars) sont locaux à chaque cellule et peuvent donc être parallélisés. L'automatisation du test de l'algorithme de diagnostic des liens du FPIN est plus complexe, car les réticules sont interconnectés. Ce test demande de déduire de la route de la chaîne JTAG le schéma des interconnexions des liens de toutes les cellules, non pas du réticule, mais du FPIN complet. Cependant, suite à une défaillance matérielle, le réticule du MiniWaferIC a dû être remplacé par une version réduite, empêchant ainsi l'application des tests à un réticule de 32×32 cellules et obligeant un travail supplémentaire d'adaptation de l'environnement de test à ce nouveau module. Cependant, seule la taille de ce réticule de 8×7 cellules diffère dans le cas de notre analyse.

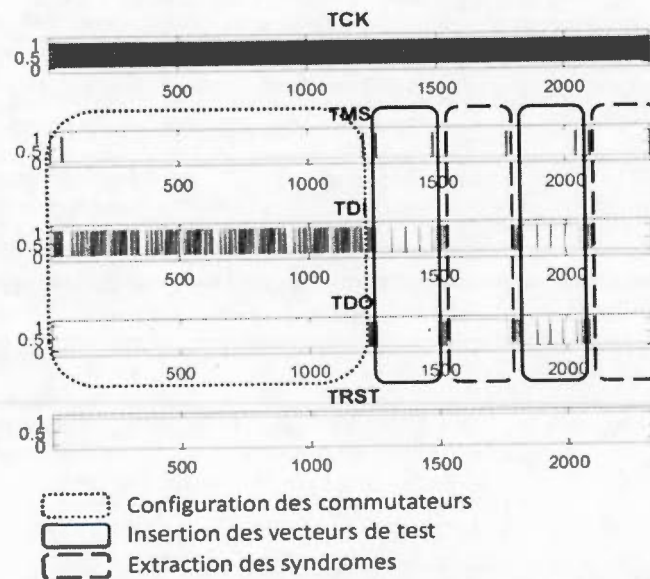


Figure 4.13 Signaux JTAG de l'application d'un test sur 4 cellules du diagnostic des commutateurs du FPIN

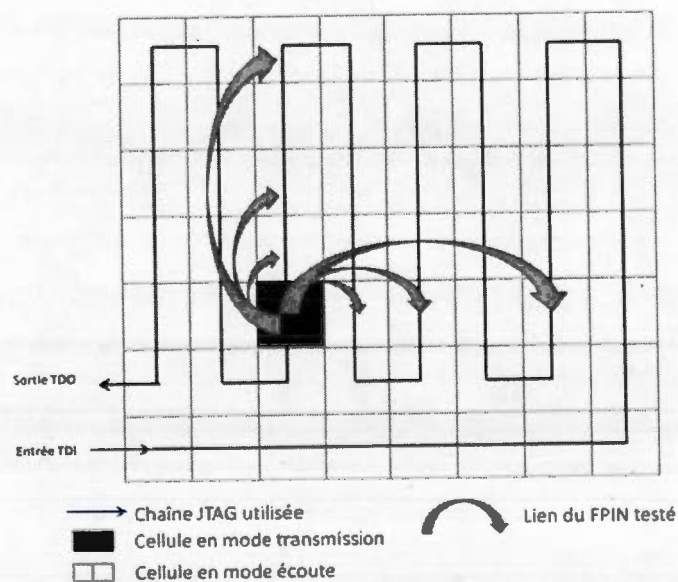


Figure 4.14 Visualisation des test de l'algorithme de diagnostic des liens du FPIN

Ainsi, le test de l'algorithme de diagnostic des liens de la figure 4.14 a été effectué pour valider son fonctionnement et se passe en trois étapes, tel que décrit à la section 3.2 :

1. La configuration des cellules, dont l'une est mise en mode transmission tandis que les autres sont en mode écoute.
2. L'insertion du vecteur de test de la cellule en mode transmission.
3. L'extraction des syndromes de toutes les cellules à l'exception de celle en mode transmission.

Afin d'appliquer le test du diagnostic des liens, une configuration des cellules a été faite, présentée à la figure 4.14. Un premier vecteur de test nécessaire à l'algorithme du « *walking one* » a ensuite été inséré dans le registre TCI pour qu'il stimule les liens de longueur 1 dans les directions Est et Nord. Suite à ce stimulus, les syndromes résultants ont été extraits. Puis, un décalage du vecteur de test a été fait, par l'introduction d'un zéro dans le registre TCI, pour qu'il stimule les liens de longueur 2 dans les mêmes directions, et les syndromes de ce nouveau stimulus ont été extraits. Enfin un nouveau décalage du vecteur de test a été fait pour stimuler les liens de longueur 4 dans les mêmes directions et les syndromes ont été extraits. L'analyse des syndromes extraits des registres TCO récupérée lors des trois tests a montré une transmission correcte des stimuli vers les cellules cibles et aucun court-circuit ou collage n'a été détecté. Le test a été appliqué sur plusieurs cellules avec des résultats identiques. Ces tests semblent valider le fonctionnement du test de l'algorithme de diagnostic des liens.

Pour permettre la comparaison du nombre de cycles théoriques et pratiques, la génération des signaux du contrôleur JTAG a été faite, sans pouvoir l'appliquer sur le réseau de 32×32 cellules, mais considérée comme valide selon les tests sur le réseau de 8×7 cellules. Le nombre de cycles d'horloge de la machine à états finis du contrôleur JTAG générés pour le test d'un réseau 32×32 cellules est de

53 642 816 cycles. La valeur théorique était estimée à 43,4 millions de cycles. Ainsi l'approximation du calcul théorique négligeant le déplacement entre les états de la machine à états finis du contrôleur JTAG a subi une augmentation de 23,6%. Cette augmentation était prévue, cependant, on remarque que l'utilisation de vecteur court de configuration (quelques cellules au lieu de toutes les cellules du réseau) tend à augmenter le pourcentage du temps de transition dans la machine à états finis du contrôleur JTAG.

4.4 Conclusion

Ce chapitre a présenté l'environnement de test réalisé pour tester et valider le prototype du WaferIC. Ce prototype valide a par la suite permis d'appliquer les algorithmes de diagnostics proposés grâce à l'outil logiciel conçu à cet effet. Une liste d'implication dans les différentes étapes de l'élaboration de l'environnement de test est présentée dans le tableau 4.2 afin de présenter le travail réalisé par les personnes impliquées. Alors que Safa Berrima s'est concentrée sur le diagnostic de la chaîne JTAG, Sylvain Charasse et moi-même avons réalisé la mise en place du banc de test. De plus, la réalisation du logiciel de diagnostic, de même que son algorithme, ont été mes tâches dédiées, comme présenté dans ce mémoire.

Une première fonction de l'outil logiciel développé a permis de configurer avec succès les chaînes JTAG reconfigurable et tolérante aux pannes. Cette fonction a aussi été utilisée pour le diagnostic des chaînes JTAG par (Berrima, 2015).

L'utilisation de cette chaîne JTAG configurée a permis d'appliquer le test de diagnostic des crossbars du FPIN localement à chaque cellule, et donc de paralléliser ce test. Ce test appliqué à un réseau de 32×32 cellules, demande 7 376 278 cycles d'horloge de la machine à états finis du contrôleur JTAG. Ce nombre de cycles est supérieur de 13.5 % de la valeur théorique estimée au chapitre 3. Ce nombre de cycles adapté à la fréquence d'horloge cible de 50 MHz pour des modules JTAG du WaferIC, représente une durée de test de 0.147 seconde.

Tableau 4.2 Liste des contributions

| | (Berrima, 2015) | (Charasse, 2013) | Gontran Sion |
|--|--------------------|---------------------|-----------------|
| Connexions d'alimentation et de communication au MiniWaferIC | 0% | 50% | 50% |
| Test et programmation du FPGA | 0% | 20% | 80% |
| Alimentation du MiniWaferIC | 0% | 50% | 50% |
| Caractéristique thermique du MiniWaferIC | 0% | 90% | 10% |
| Outil logiciel pour l'analyse et l'automatisation | 0% | 0% | 100% |
| Diagnostic des chaîne JTAG | 100% | 0% | 0% |
| Diagnostic du FPIN | 0% | 0% | 100% |

De plus, le test de diagnostic des liens du FPIN a été appliqué sur un réseau de 8×7 cellules, et a montré sa validité. Cependant, il n'a pu être appliqué sur un réseau de 32×32 cellules, car il a été endommagé au cours de test. Le nombre de cycles d'horloge de la machine à états finis du contrôleur JTAG générés par l'outil de test pour effectuer ce diagnostic sur un réseau de 32×32 cellules est de 53 642 816 cycles, soit une augmentation de 23.6% par rapport à la valeur estimée théorique. Cette valeur est 10.4 fois inférieure à celle de l'algorithme proposé par Basile par (Basile-Bellavance, 2009), qui était de 5.6 Gcycles. Le nombre de cycles nécessaire à l'algorithme proposé appliqué à la fréquence d'horloge cible de 50 MHz des modules JTAG représente une durée d'application du test de 1.07 seconde.

Ainsi l'algorithme proposé permet l'application du test de diagnostic des *crossbars* et des liens du FPIN en 61 019 094 cycles, soit un temps de test à 50 MHz de

1.22 seconde, alors que celui proposé par (Basile-Bellavance, 2009) nécessitait 5.61 G cycles, soit une durée de test à 50 MHz de 112.2 secondes.

CHAPITRE V

CONCLUSION

Le projet de recherche DreamWafer™, regroupant plusieurs universités canadiennes, de même que des partenaires industriels, visait à élaborer une plateforme de prototypage rapide pour les systèmes numériques, permettant de concevoir un système électronique en quelques heures au lieu de jours ou mois. Ce dispositif est comparable à un circuit imprimé reprogrammable et intelligent, permettant d'interconnecter les composants électroniques déposés à sa surface conformément aux spécifications fournies. Le WaferIC™, au cœur de cette plateforme de prototypage, est un circuit intégré de très grande surface (*Very Large Area Integrated Circuit*, VLAIC) dont la surface est composée d'une matrice de contacts, les NanoPads, permettant de connecter les composants électroniques à sa surface au réseau d'interconnexions programmables (*Field Programmable Interconnect Network*, FPIN). Ce circuit intégré à l'échelle de la tranche de silicium (*Wafer Scale Integrated Circuit*, WSIC) de 200 mm de diamètre est composé de 76 réticules identiques répétés sur toute la surface. Ces réticules sont eux-mêmes composés d'une mer de 32×32 cellules identiques et interconnectées formant le FPIN. D à la surface de ce circuit intégré, les pannes sont inévitables et un diagnostic du FPIN doit être effectué afin de permettre d'appliquer une stratégie de tolérance aux pannes en les contournant.

L'objectif principal de cette recherche était d'augmenter la couverture des pannes et de réduire le temps de test du précédent algorithme de diagnostic du FPIN proposé

par (Basile-Bellavance, 2009), ainsi que d'implémenter le test du diagnostic proposé sur le prototype physique qu'est le MiniWaferIC. Ce prototype du WaferIC développé par l'équipe DreamWafer, est composé de 2×2 réticules connectés à une structure de contrôle et d'alimentation, le PowerBlock. **(contribution 1)** Un environnement de test a ainsi été développé conjointement avec l'étudiant Sylvain Charasse pour permettre la validation du fonctionnement du prototype. Afin de permettre le contrôle et la configuration du FPIN en vue du diagnostic, **(contribution 2)** un outil logiciel a été développé. Le WaferIC dispose d'une chaîne JTAG reconfigurable et tolérante aux pannes, qui demande une première configuration avant toute possibilité d'accès aux registres des cellules du FPIN. Ainsi, l'outil logiciel permet la configuration d'une chaîne JTAG passant par les cellules désirées. **(contribution 3)** Cette seule fonctionnalité a permis de valider les tests de l'algorithme de diagnostic de la chaîne JTAG élaboré par (Berrima, 2015). Enfin suite à la construction de la chaîne JTAG, **(contribution 4)** l'outil logiciel a permis, l'accès à tous les registres des cellules, et de configurer les registres nécessaires aux tests du diagnostic proposé.

L'algorithme de diagnostic proposé, qui a fait l'objet **(contribution 5)** d'une publication à la conférence IOLTS 2015 (Sion *et al.*, 2015) (Appendice A), se divise en deux séries de tests distincts s'appliquant, l'une sur les *crossbars* programmables des cellules qui sont la partie active du FPIN, et l'autre sur les liens du réseau d'interconnexions reliant toutes les cellules entre-elles.

Les *crossbars* programmables permettent le routage du FPIN en connectant les liens entrants d'une cellule à ses liens sortants. La configuration des commutateurs du crossbar s'effectue par la chaîne JTAG en accédant aux registres de configuration CR de chaque cellule. Chaque entrée du *crossbar* est testée par une configuration particulière connectant l'entrée sous test à toutes les sorties, nommé configuration en mode *broadcast*. Pour chacune des configurations, le crossbar est testé par deux

vecteurs complémentaires insérés par la chaîne JTAG dans les registres TCI et en récupérant les syndromes des registres TCO. Le fait que les tests des *crossbars* soient locaux permet de les paralléliser. L'algorithme de diagnostic des *crossbars* programmables proposé dans ce mémoire est en principe le même que celui proposé par (Basile-Bellavance, 2009), et seule l'application le différencie de par l'architecture différente des registres de configuration, d'insertion et d'extraction des commutateurs. La programmation des commutateurs demande 142 bits par *crossbars* plutôt que 314 dans l'architecture utilisée par Basile dans (Basile-Bellavance, 2009), soit une réduction de 55 % du temps de configuration des *crossbars* programmables. Ainsi, il a été démontré que pour le test du diagnostic des *crossbars* d'un réseau de 32×32 cellules, **(contribution 6)** une réduction de la durée du test par un facteur de 1.8 a été obtenue par rapport à celle de (Basile-Bellavance, 2009), faisant passer le nombre de cycles nécessaires à la machine à états finis du contrôleur JTAG de 11.8 M à 6.6 M par rapport à l'algorithme de (Basile-Bellavance, 2009).

Les liens étant de longueur allant jusqu'à 32 cellules et traversant tout le réseau, le test du diagnostic demande la définition d'un cône d'influence afin d'éviter l'interférence des tests. **(contribution 7)** L'algorithme de diagnostic des liens proposé tire avantage de l'architecture du FPIN pour optimiser le cône d'influence de l'algorithme existant. Les liens d'une cellule étant soit horizontaux, soit verticaux, le cône d'influence est réduit aux seules cellules de la même ligne, horizontale et verticale, distantes de moins de 32 cellules (Fig. 3.5). Ce cône d'influence permet la diagonalisation du test. De plus, il permet de tester tous les réseaux parallèlement avec la même configuration et les mêmes vecteurs de test. Cette diagonalisation a permis de tester N cellules en parallèle pour chacun des réseaux de $N \times N$ cellules. Le fait de tester N cellules à la fois empêche N-1 cellules de la diagonale de tester leurs liens avec la cellule restante de cette diagonale. Il a été ainsi proposé de tester les diagonales et leurs perpendiculaires afin de compléter la couverture des pannes. **(contribution 8)** De plus, en remarquant que le test de 2 liens simultanés et dans

deux directions perpendiculaires peut être fait par l'algorithme du « walking one » et « walking zero » sans recouvrement du cône d'influence, cela permet de réduire le nombre de tests par deux. En résumé, **(contribution 9)** l'algorithme diagonalisé du diagnostic des liens du FPIN permet de réduire le nombre de tests de $O(n^4)$ à $O(n^3)$, où n est le nombre de liens par rapport à l'algorithme proposé par (Basile-Bellavance, 2009). Il a aussi été montré que la durée du test du diagnostic des liens du FPIN d'un réseau de 32×32 cellules a été réduite par un facteur de 129 avec l'algorithme proposé, faisant passer le nombre de cycles nécessaires à la machine à état fini du contrôleur JTAG de 5.6 G.à 43.4 M par rapport à l'algorithme de (Basile-Bellavance, 2009).

Enfin, il a été montré dans ce mémoire que l'algorithme de diagnostic des *crossbars* programmables et des liens du FPIN permettait, pour une couverture complète du modèle de pannes simple des collages et courts-circuits, une réduction de la durée de test de deux ordres de grandeur par rapport à l'algorithme précédent, faisant passer le test complet d'un réseau de 32×32 cellules de 112.2 secondes à 1.22 seconde à une fréquence d'horloge du contrôleur JTAG de 50 MHz.

Améliorations futures :

Le programme présenté au chapitre 4 permet l'insertion et l'extraction de vecteurs de test et de syndrome, et la partie diagnostic n'a pas été complètement implémentée. Ainsi le système complet pourrait être automatisé.

L'extraction des syndromes pour l'algorithme de diagnostic des *crossbars*, mais surtout des liens est la partie la plus chronophage. Advenant un besoin de réduire la durée des tests sous les secondes, l'implémentation d'un analyseur de syndromes (*output response analyser*, ORA) *in situ*, serait une piste à envisager pour réduire drastiquement le temps de test et diagnostic, au prix d'une surface accrue du matériel de test dans le circuit intégré.

Enfin, le test de diagnostic des pannes de délai sur les liens d'interconnexion s'avérerait important pour des systèmes qui demandent une fréquence de fonctionnement élevée. Ainsi, de nouvelles structures pourraient être ajoutées dans les cellules, comme les PLL ou des lignes à délai programmables, afin de couvrir les pannes de délais. Une autre piste serait d'utiliser un FPGA déposé sur le WaferIC, par exemple, afin de générer les tests du FPIN de sa surface.

BIBLIOGRAPHIE

- Abraham, J.A. et Fuchs, W.K. (1986). Fault and error models for VLSI. *Proceedings of the IEEE*, 74(5), 639-654. doi: 10.1109/PROC.1986.13528
- Al-Arian, S.A. et Agrawal, D.P. (1987). Physical failures and fault models of CMOS circuits. *Circuits and Systems, IEEE Transactions on*, 34(3), 269-279. doi: 10.1109/TCS.1987.1086138
- André, W., Blaqui re, Y. et Savaria, Y. (2011). *A Wafer-Scale Rapid Electronic Systems Prototyping Platform*. : INTECH Open Access Publisher.
- Aviziens, A. (1976). Fault-Tolerant Systems. *Computers, IEEE Transactions on*, C-25(12), 1304-1312. doi: 10.1109/TC.1976.1674598
- Bandler, J.W. et Salama, A.E. (1985). Fault diagnosis of analog circuits. *Proceedings of the IEEE*, 73(8), 1279-1325. doi: 10.1109/PROC.1985.13281
- Banerjee, P. et Abraham, J.A. (1984). Characterization and Testing of Physical Failures in MOS Logic Circuits. *Design & Test of Computers, IEEE*, 1(3), 76-86. doi: 10.1109/MDT.1984.5005655
- Basile-Bellavance, Y. (2009). Faults diagnosis methodology for the WaferNet interconnection network. Circuits and Systems and TAISA Conference, 2009. NEWCAS-TAISA '09. Joint IEEE North-East Workshop on.
- Basile-Bellavance, Y. (2010). *Conception d'un systeme de test et de configuration numerique tolerant aux pannes pour la technologie waferic*. (M.Sc.A.). Ecole Polytechnique, Montreal (Canada), Ann Arbor.

- Berrima, S. (2015). *Algorithmes de diagnostic d'une chaîne JTAG reconfigurable et tolérante aux pannes au sein de la technologie WaferIC*. (M.Sc.A.). École Polytechnique de Montréal.
- Blaquiere, Y., Basile-Bellavance, Y., Berrima, S. et Savaria, Y. (2014, 1-5 June 2014). Design and validation of a novel reconfigurable and defect tolerant JTAG scan chain. *Circuits and Systems (ISCAS)*, 2014 IEEE International Symposium on (p. 2559-2562).
- Burgess, N. et Damper, R.I. (1984). The inadequacy of the stuck-at fault model for testing mos lsi circuits: a review of mos failure mechanisms and some implications for computer-aided design and test of mos lsi circuits. *Software & Microsystems*, 3(2), 30-36. doi: 10.1049/sm:19840011
- Case, G.R. (1976). Analysis of actual fault mechanisms in CMOS logic gates. *Proceedings of the 13th Design Automation Conference* (p. 265-270). San Francisco, California, USA : ACM
- Charasse, S. (2013). *Outil de vérification in-situ pour le prototypage de systèmes électroniques*. École Polytechnique de Montréal.
- Chiang, K.-W. et Vranesic, Z.G. (1983). On fault detection in CMOS logic networks. *Proceedings of the 20th Design Automation Conference* (p. 50-56). Miami Beach, Florida, USA : IEEE Press
- Corsi, F. (1991). Inductive fault analysis revisited [integrated circuits]. *Circuits, Devices and Systems, IEE Proceedings G*, 138(2), 253-263.
- Courtois, B. (1981, 1981). Failure mechanisms, fault hypotheses and analytical testing of LSI-NMOS (HMOS) circuits. *VLSI-81.-Very-Large-Scale-Integration.-Proceedings-of-the-First-International-Conference-on-Very-Large-Scale-Integration*. : Academic Press, London, UK. Récupéré de <http://hal.archives-ouvertes.fr/hal-00013576>

- Dick, J. (2015) *Apple Watch and ASE Start New Era in SiP*. de <http://www.chipworks.com/about-chipworks/overview/blog/apple-watch-and-ase-start-new-era-sip>
- Doumar, A. et Ito, H. (2003). Detecting, diagnosing, and tolerating faults in SRAM-based field programmable gate arrays: a survey. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(3), 386-405. doi: 10.1109/TVLSI.2002.801609
- Elzig, Y.M. (1981, 29-1 June 1981). Automatic Test Generation for Stuck-Open Faults in CMOS VLSI. Design Automation, 1981. 18th Conference on (p. 347-354).
- Fault Modeling. (1985). *Design & Test of Computers, IEEE*, 2(2), 88-95. doi: 10.1109/MDT.1985.294873
- FTDI. (2011) *UM232R USB - Serial UART Development Module*. de http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM232R.pdf
- Furber, S. et Bainbridge, J. (2005, 17-17 Nov. 2005). Future Trends in SoC Interconnect. System-on-Chip, 2005. Proceedings. 2005 International Symposium on (p. 183-186).
- Gai, S., Mezzalana, M. et Prinetto, P. (1983). A review of fault models for lsi/vlsi devices. *Software & Microsystems*, 2(2), 44-53. doi: 10.1049/sm:19830016
- Galiay, J., Crouzet, Y. et Vergniault, M. (1980). Physical Versus Logical Fault Models MOS LSI Circuits: Impact on Their Testability. *Computers, IEEE Transactions on*, C-29(6), 527-531. doi: 10.1109/TC.1980.1675614
- Hassan, A., Rajski, J. et Agarwal, V.K. (1988, 12-14 Sep 1988). Testing and diagnosis of interconnects using boundary scan architecture. Test Conference, 1988. Proceedings. New Frontiers in Testing, International (p. 126-137).

Huang, W.K., Chen, X.T. et Lombardi, F. (1996, 28 Apr-1 May 1996). On the diagnosis of programmable interconnect systems: Theory and application. VLSI Test Symposium, 1996., Proceedings of 14th (p. 204-209).

IEEE Standard Test Access Port and Boundary Scan Architecture. (2001). *IEEE Std 1149.1-2001*, 1-212. doi: 10.1109/IEEESTD.2001.92950

Jarwala, N. et Chi, W.Y. (1989, 29-31 Aug 1989). A new framework for analyzing test generation and diagnosis algorithms for wiring interconnects. Test Conference, 1989. Proceedings. Meeting the Tests of Time., International (p. 63-70).

Kautz, W.H. (1974). Testing for Faults in Wiring Networks. *Computers, IEEE Transactions on*, C-23(4), 358-363. doi: 10.1109/T-C.1974.223950

Lala, P.K. (2009). *An Introduction to Logic Circuit Testing*. : Morgan & Claypool Publishers.

(dir.). (1995, 27-30 Jun 1995). *DEPENDABLE COMPUTING AND FAULT TOLERANCE : CONCEPTS AND TERMINOLOGY*. Fault-Tolerant Computing, 1995, Highlights from Twenty-Five Years., Twenty-Fifth International Symposium on.

Mangir, T.E. (1984). Sources of failures and yield improvement for VLSI and restructurable interconnects for RVLSI and WSI: Part I—Sources of failures and yield improvement for VLSI. *Proceedings of the IEEE*, 72(6), 690-708. doi: 10.1109/PROC.1984.12917

McCracken, S. et Zilic, Z. (2002). FPGA test time reduction through a novel interconnect testing scheme. Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays (p. 136-144). Monterey, California, USA : ACM

Meng, L., Savaria, Y., Bing, Q. et Taillefer, J. (2003, 3-5 Nov. 2003). IEEE 1149.1 based defect and fault tolerant scan chain for wafer scale integration. Defect

and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on (p. 18-25).

Modeling and optimization of SiP/SoP and packaging for electrical integrity. (2010).
de
https://www.cst.com/content/events/workshop_documents/2010/Modeling_and_optimization_of_SiPSoP_and_packaging_for_electrical_integrity.pdf

Norman, R. (2008). *Reprogrammable circuit board with alignment-insensitive support for multiple component contact types*. U.S.Patent 8 124 429.

Norman, R., Valorge, O., Blaquiere, Y., Lepercq, E., Basile-Bellavance, Y., El-Alaoui, Y., Prytula, R. et Savaria, Y. (2008, 22-25 June 2008). An active reconfigurable circuit board. Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop on (p. 351-354).

Poupon, G., Sillon, N., Henry, D., Gillot, C., Mathewson, A., Di Cioccio, L., Charlet, B., Leduc, P., Vinet, M. et Batude, P. (2009). System on Wafer: A New Silicon Concept in SiP. *Proceedings of the IEEE*, 97(1), 60-69. doi: 10.1109/JPROC.2008.2007464

RAO, T. (2006) *SoC vs. MCM vs SiP vs. SoP.* de
<http://electroiq.com/blog/2006/07/soc-vs-mcm-vs-sip-vs-sop/>

Renovell, M., Figueras, J. et Zorian, Y. (1997, 27 Apr-1 May 1997). Test of RAM-based FPGA: methodology and application to the interconnect. VLSI Test Symposium, 1997., 15th IEEE (p. 230-237).

Renovell, M., Portal, J.M., Figueras, J. et Zorian, Y. (1998). Testing the interconnect of RAM-based FPGAs. *Design & Test of Computers, IEEE*, 15(1), 45-50. doi: 10.1109/54.655182

Shen, J.P., Maly, W. et Ferguson, F.J. (1985). Inductive Fault Analysis of MOS Integrated Circuits. *Design & Test of Computers, IEEE*, 2(6), 13-26. doi: 10.1109/MDT.1985.294793

- Sion, G., Blaquiere, Y. et Savaria, Y. (2015). Defect diagnosis algorithms for a field programmable interconnect network embedded in a Very Large Area Integrated Circuit. On-Line Testing Symposium (IOLTS), 2015 IEEE 21st International.
- Sying-Jyan, W. et Chao-Neng, H. (1998, 2-4 Dec 1998). Testing and diagnosis of interconnect structures in FPGAs. Test Symposium, 1998. ATS '98. Proceedings. Seventh Asian (p. 283-287).
- Wadsack, R.L. (1978). Fault modeling and logic simulation of CMOS and MOS integrated circuits. *AT T Technical Journal*, 57, 1449-1474. Récupéré de <http://adsabs.harvard.edu/abs/1978ATTTJ..57.1449W>
- Wang, L.T., Wu, C.W. et Wen, X. (2006). *VLSI Test Principles and Architectures: Design for Testability*. : Elsevier Science.
- Yang, C., Huang, L. et Zhu, M. (2010, 15-17 June 2010). Study on test generation and diagnosis optimization algorithm for wiring interconnects. Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on (p. 555-559).
- Yinlei, Y., Jian, X., Wei Kang, H. et Lombardi, F. (1999a, 18-21 Jan 1999). Diagnosing single faults for interconnects in SRAM based FPGAs. Design Automation Conference, 1999. Proceedings of the ASP-DAC '99. Asia and South Pacific (p. 283-286 vol.281).
- Yinlei, Y., Jian, X., Wei Kang, H. et Lombardi, F. (1999b, 1999). Minimizing the number of programming steps for diagnosis of interconnect faults in FPGAs. Test Symposium, 1999. (ATS '99) Proceedings. Eighth Asian (p. 357-362).
- Yu, Q. et Ampadu, P. (2012). Transient and Permanent Error Control for Networks-on-Chip. Dans Springer (dir.),

Yue, W. et Dongfang, L. (2002, 4-7 Aug. 2002). A fast diagnosis method for interconnect fault in FPGA. Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on (p. III-231-234 vol.233).

APPENDICE A

ARTICLE DE CONFÉRENCE

Sion, G., Blaquiere, Y., et Savaria, Y. (2015). Defect diagnosis algorithms for a field programmable interconnect network embedded in a very large area integrated circuit. On-Line Testing Symposium (IOLTS), 2015 IEEE 21st International.(p. 83-88).

Defect Diagnosis Algorithms for a Field Programmable Interconnect Network Embedded in a Very Large Area Integrated Circuit.

Gontran Sion, Yves Blaquière

Computer Science Department, CoFaMiC

Université du Québec à Montréal

Montréal, Canada

gontran.sion@gmail.com, blaquiere.yves@uqam.ca

Yvon Savaria

GR2M, Electrical Engineering Department

École polytechnique Montréal

Montréal, Canada

yvon.savaria@polymtl.ca

Abstract— Algorithms are proposed to diagnose defects in a defect tolerant field programmable interconnection network embedded in a large area integrated circuit. The proposed diagnosis algorithms use a diagonal configuration approach to reduce the cone of influence of individual tests, thus allowing parallel tests according to diagonal patterns. The proposed algorithms avoid redundant diagnosis tests. Efficiency of the proposed diagnosis algorithms are calculated in terms of the number of cycles of a JTAG FSM required to apply the test. Results show a 113-fold test time reduction in the considered interconnection network.

Keywords— Test and diagnosis, field programmable interconnection network (FPIN), defect tolerant, very large area integrated circuit (VLAIC).

I. INTRODUCTION

Defect diagnosis attempts to locate defective and functional elements in integrated circuits. Results can be used to improve device yield, by statically or dynamically repairing circuits that comprise defect tolerant structures [1].

Another application relates to imperfect circuits rejected by full functionality tests, but that can be used in other applications. For example, Xilinx runs defect diagnosis on partly defective rejected SRAM-based Field Programmable Gate Arrays (FPGAs) to harvest devices that are offered as the EasyPath™ product to their customers [2, 3]. Rejected FPGA samples are diagnosed to extract their respective defects maps. Those compatible with some specific end-user production application (bit-file) are offered at lower cost.

Any Very Large Area Integrated Circuit (VLAIC) exploiting defect tolerance techniques [4] based on field programmable technologies can adopt a configuration strategy based on defect maps [1]. VLAICs are typically made with photo-repetition of reticule images interconnected with reticule stitching techniques [5]. This technique could be used to make very large Programmable Logic Devices, Interconnection on Chip (IoC) or active interposers [6, 7].

Configuration based on defect maps was used in a rapid prototyping platform for electronic systems [8]. This platform

is based on a wafer-scale active reconfigurable circuit that is used to interconnect components deposited on its surface.

As in FPGAs, this reconfigurable circuit is composed of two types of structures, the functional part and the configurable interconnect network. The functional part has an active surface made of a dense array of very fine conducting pads called NanoPads [Fig. 1]. These NanoPads can detect and make contact with pins of integrated circuit components. The corresponding electrical signals can then be dynamically connected to others by configuring a defect tolerant field programmable mesh interconnection network, called FPIN [9], made of crossbars.

A diagnosis methodology for the FPIN was introduced in [10]. A sequential walking one algorithm and a broadcast algorithm were proposed to locate single shorts or stuck at faults in interconnects whereas only stuck at were diagnosed in the crossbars, and no short diagnosis in the crossbars was performed. Further analysis of this diagnosis method led to the observation that the number of syndromes to extract and the broad cone of influence of each test can make the diagnosis time very long.

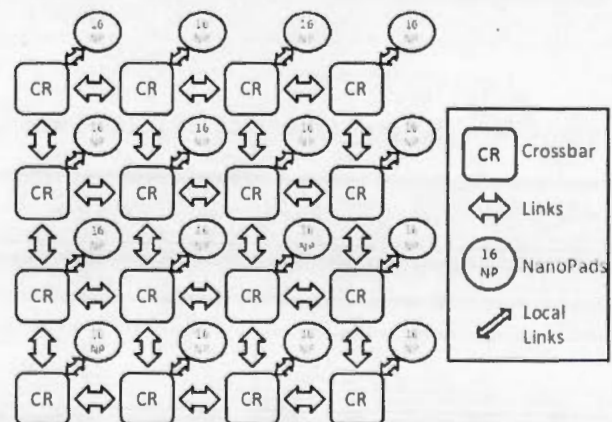


Fig. 1. Simplified structure of the target FPIN.

This paper proposes a set of new and improved defect diagnosis algorithms for the FPIN that cover single stuck at and short defects for both interconnects and crossbars. These exhaustive and deterministic algorithms offer a better defect coverage and are in the worst case 113 times faster than the algorithm proposed in [10].

The rest of the paper first details the FPIN architecture and the proposed diagnosis algorithm in Section II. Section III presents results on the diagnosis time complexity while section IV reports results of experimental tests on the FPIN. Finally, conclusions are summarized in Section V.

II. FPIN ARCHITECTURE AND PROPOSED DIAGNOSIS ALGORITHMS

Diagnosis of the FPIN may either extract a map of defects, or an inverse map of proven functional elements. These maps can then subsequently be used by placement and routing algorithms to bypass defective elements while configuring a VLAIC [11] or an FPGA. More effective defect diagnosis algorithms can be developed by exploiting knowledge on an architecture. Thus, relevant information on the architecture and its structure are summarized next.

A. Architecture of the defect tolerant configurable VLAIC

The VLAIC implemented in the rapid prototyping platform for electronic systems [8] is made of reticle images of 32×32 cells. Each cell incorporates 4×4 NanoPads connected to a crossbar, as shown in Fig. 2. A cell is configured or tested through four independent scan chains dedicated to: NanoPad Configuration (NC), Test Crossbar Input (TCI), Test Crossbar Output (TCO) and Crossbar (CR) chains. For example, the NanoPad chain includes multiple registers to configure NanoPads and their muxes. These chains are connected to a reconfigurable and defect tolerant JTAG (Joint Test Action Group) chain [12], with one Test Access Port (TAP) controller per cell (not shown in Fig. 2). Each reticle has its own JTAG port and can also be configured from neighbor reticle JTAG ports, which allows defect tolerance and parallel configuration.

Each crossbar has 6 incoming links and 6 outgoing links with respective lengths of 1, 2, 4, 8, 16, and 32 cells, in the 4 directions North, South, East, and West [Fig. 3]. Each crossbar has also 2 links coming from two of the 16 NanoPads and 4 links going to two of the 16 NanoPads, for a total of 26 incoming and 28 outgoing links. The crossbar is therefore implemented with 28 26-to-1 multiplexers. The 26 bit crossbar input can be bypassed by a 26 bit TCI register to test the crossbar. Both the crossbar core and its input mux are configured by the CR register. So the FPIN diagnostic configuration is performed through the 142 bit CR register. Finally, the crossbar core output can be sampled to a 28 bit TCO register, which allows syndromes extraction [Fig. 2].

B. Proposed diagnosis algorithms

The proposed algorithms diagnose defects in the defect tolerant field programmable interconnection network in two phases. The crossbars are first diagnosed, and then the FPIN links.

1) Crossbar Diagnosis Algorithm

Crossbar diagnosis is local to each cell and can therefore be performed in parallel. As proposed in [10], each of the 26 crossbar's input are successively configured in broadcast mode (one-to-all configuration) and the diagnosis algorithm shown in Fig. 4 is applied on it. A sequence of broadcast configuration, test generation and syndrome extraction is applied for each input.

Unlike in [10] where a walking (one and zero) algorithm was applied on a 26-bit register, which requires the extraction of 52 syndromes for each one-to-all configuration, just 2 vectors are applied [Fig. 4, steps 2 and 2']. The 2 vectors are a '1' on the active signal for each broadcast crossbar configuration while the others are at '0', and the complement. For each applied vector, a syndrome is extracted [Fig. 4, steps 3 and 3'] to be analyzed for defect diagnosis.

These two test vectors are applied for each active signal and each configuration, and are sufficient to diagnose all single stuck-at and short faults in the crossbar I/Os like the walking test proposed in [10]. It divides the number of syndrome extractions for all configurations by 26. For a full 32×32 cells reticle, it reduces the number of cycles from 38.5 M cycles to 1.5 M cycles. While these syndromes can directly locate any stuck-at faults when compared with expected fault-free syndrome, analysis of syndromes combinations could be required to precisely locate single shorts.

As shown in the example of a 3×3 crossbar in Fig. 5, tested stuck-at can be at input (b) or output (c) of crossbars and tested shorts can be at input (d), output (e), or between inputs and outputs (f) of crossbars. In Fig. 5, column (a) is fault free and (C1), (C2), and (C3) are the three possible broadcast configurations of a 3×3 crossbar.

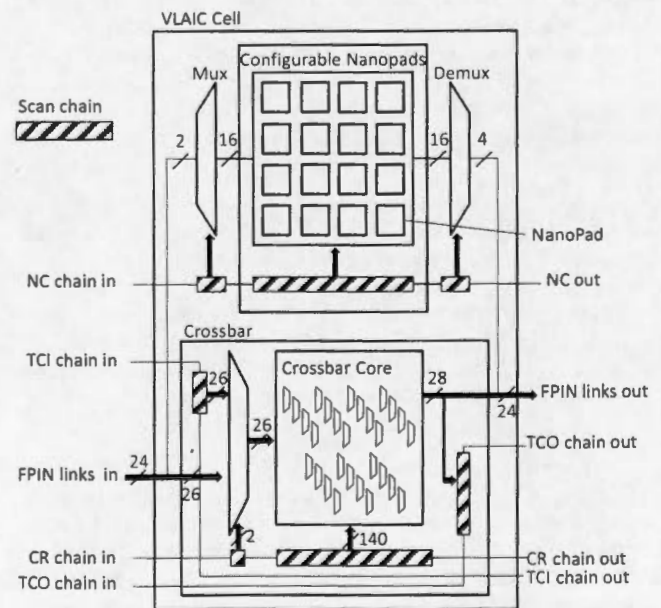


Fig. 2. Architecture of the VLAIC cell related to FPIN.

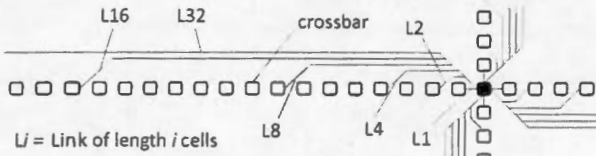


Fig. 3. Crossbar link connections in the FPIN. Only connections from one crossbar (black box) to the West side crossbars in one direction are shown.

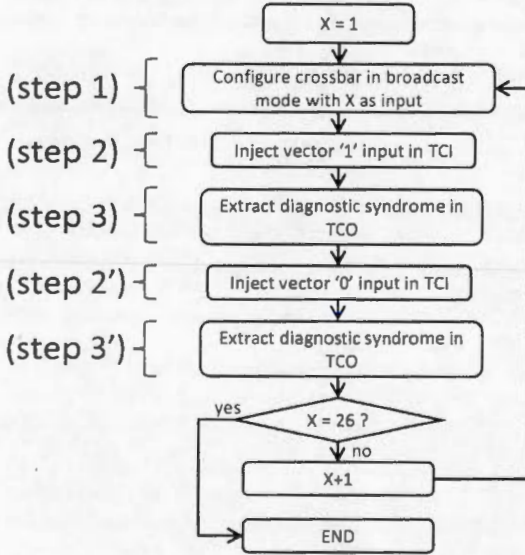


Fig. 4. FPIN Crossbar Diagnosis Algorithm.

As shown in Table 1, the $2N$ (with $N=3$) test vectors applied to the 3×3 crossbar would allow diagnosing the 5 distinct representative faults. Indeed Fig. 5 reports the resulting distinct syndromes for single stuck at and short faults, except for the syndrome resulting from output shorts that are implicitly diagnosed in the FPIN link diagnosis algorithm. Different syndromes show how various faults can be diagnosed.

For example, if the six test vectors of Table 1 are applied to a stuck-at fault input of a crossbar (Fig 5 column (b)), the (b) syndromes in Table 1 are observed rather than the fault free (a) syndromes. In this case, only the test vector "101" diagnoses the stuck-at one defect on the second input, as the second input configured in broadcast mode (C2) creates an output (b) syndrome "111" different (identified with shaded grey in Table 1) from the expected "000" fault-free (a) syndrome.

2) FPIN Links Diagnosis Algorithm

In the diagnosis algorithm proposed in [10], a complex and large cone of influence (Fig. 6) was defined, and this reduced the number of parallelized cell test per reticule to 1 or 2. Limiting the size of the cone of influence prevents a test from disturbing another one. The cone of influence defines a security gap of 2 times the size of the longest (L_{max}) tested link.

Based on knowledge of the topology of the FPIN links (horizontal and vertical) and the fact that the longest links are

Table 1. Syndromes produced by the various diagnosable single faults.

| Test vectors | (a) syndromes | (b) syndromes | (c) syndromes |
|-------------------------|--------------------------|-------------------------|--------------------------|
| 1 0 0 0 1 1 | 1 1 1 0 0 0 | 1 1 1 0 0 0 | 1 0 1 0 0 0 |
| 0 1 0 1 0 1 | 1 1 1 0 0 0 | 1 1 1 1 1 1 | 1 0 1 0 0 0 |
| 0 0 1 1 1 0 | 1 1 1 0 0 0 | 1 1 1 0 0 0 | 1 0 1 0 0 0 |
| (d) Syndromes OR bridge | (d) Syndromes AND bridge | (f) Syndromes OR bridge | (f) Syndromes AND bridge |
| 1 1 1 1 1 1 | 0 0 0 0 0 0 | 1 1 1 0 1 0 | 1 0 1 0 0 0 |
| 1 1 1 1 1 1 | 0 0 0 0 0 0 | 1 1 1 0 1 0 | 1 0 1 0 0 0 |
| 1 1 1 0 0 0 | 1 1 1 0 0 0 | 1 1 1 0 0 0 | 1 1 1 0 0 0 |

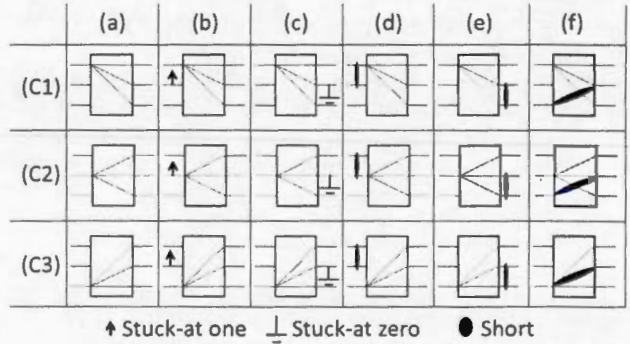


Fig. 5. 3×3 crossbar single fault models.

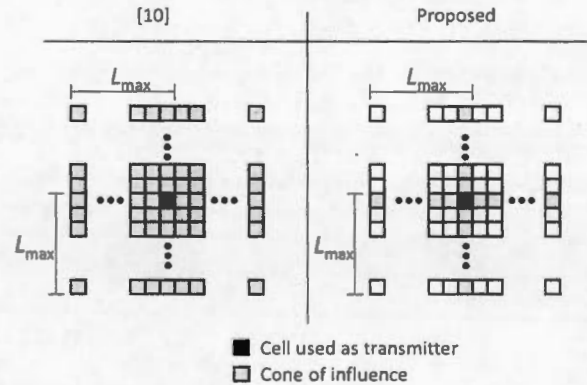


Fig. 6. Cone of influence of the proposed FPIN Diagnosis algorithm and of the one presented in [10].

the size of a reticule's side, we propose to reduce the size of the cone of influence to cells that share the same horizontal and vertical position as the tested cell [Fig. 6]. This property allows testing N cells in parallel for a reticule of $N \times N$ cells. So we proposed to test each reticule with the same test applied on cells in a moving diagonal [Fig. 7].

The test is performed by first programming all cells [Fig. 7 Conf 1], except the first diagonal, in listening mode (bypass mode), which allows them to retrieve syndromes of all their incoming links. The cells of the first diagonal are programmed as test transmitters, and will stimulate links under test. Each

following configuration (Conf 2 to Conf 7 and Conf 1' to Conf 7') needs to reprogram only cells on two diagonals: the cells previously programmed as transmitters are reprogrammed in bypass mode, while the cells on the new diagonal are reprogrammed as test transmitters.

All cells of the diagonal transmit simultaneously the same test vectors. Most single shorts between two perpendicular links are diagnosable. An exception exists with short between link-under-test and link not under test that cannot be listened to, i.e. input links of transmitter cells of the same diagonal [Fig. 8(a)]. These single shorts can be located if the same test is performed with two perpendicular diagonals, which ensure that at least one cell is listened to for each link not under test, as shown in Fig. 8 where cell CC enters in listening mode in (b). So each cell is diagnosed with two complementary configurations for a full links diagnosis.

If test time is very critical, a multi-diagonal configuration mode is possible. As links have respective lengths of 1, 2, 4, 8, 16, and 32 cells (Fig. 3), each diagonal can then be separated by 2 cells without any conflict, so the number of configurations is reduced to 3 (6 with complementary diagonals) from 32 (64). The trade-off is in a reduction of the shorts diagnosability that is caused by an overlap of the cones of influence.

In [10], the walking one algorithm test consists in shifting one slice of a single one surrounded by zeros in each TCI register. This walking one is applied for each configuration to stimulate each link [Fig. 9 (2) and (2')]. Our proposed walking one algorithm takes advantage of the architecture, where two slices of '1' surrounded by '0's per TCI register allow to test one

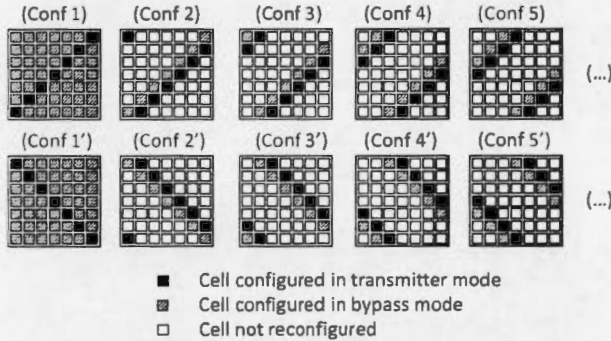


Fig. 7. FPIN diagonal diagnosis algorithm shown on a reticule of 7x7 cells.

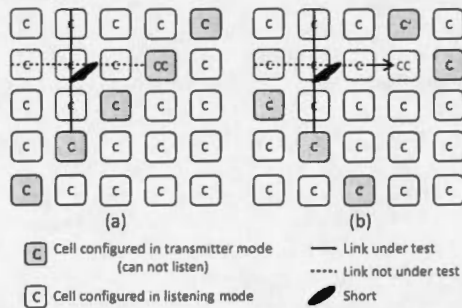


Fig. 8. Perpendicular diagonals for full single short coverage.

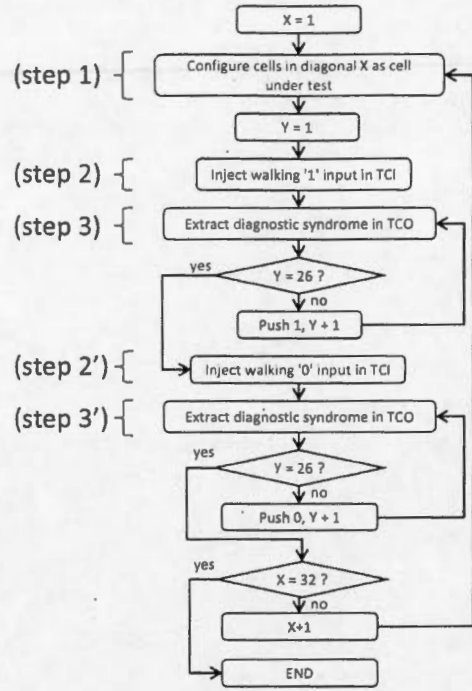


Fig. 9. FPIN links diagnosis algorithm.

algorithm takes advantage of the architecture, where two slices of '1' surrounded by '0's per TCI register allow to test links simultaneously in horizontal and vertical directions, reducing the number of shifts (push) by two.

Links could also be separated in 4 slices (North, South, East, and West), to reduce by four the number of shifts. This 4 slice scenario causes some overlap of the cones of influence and mask some short defects.

III. COMPLEXITY ANALYSIS AND RESULTS

The FPIN is implemented as a regular fabric, obtained by combining sub networks of 32x32 cells laid out over each reticule image. Each reticule also has a JTAG port that allows configuring and diagnosing each reticule in parallel. Thus, it is possible to configure, insert and extract test vectors through each reticule, in parallel (analogous to a star JTAG architecture).

For simplicity, the following test time complexity analysis does not take into account JTAG FSM control and configuration times. These times are required to pass between different JTAG FSM states. However, as it takes only few cycles for these tasks, neglecting these cycles has a small impact, and the analysis produces fairly accurate results.

A. Diagnosis time for the crossbar

According to the steps in Fig. 4, the total time measured in clock cycles to configure all crossbars [Fig. 4 (1)] is

$$T_c = (N^2 CR) TCI \quad (1)$$

where N^2 is the number of cells per reticule, CR the size of the configuration register (142 bits) and TCI the number of configurations according to the number of crossbar inputs (26).

The N^2 CR registers in a reticle are serially connected through the same JTAG user register chain.

Two complementary test vectors for each of the crossbar configurations are generated: a vector with a '1' for the crossbar link under test with the others at '0', and the complement. The total time for the injection of the two test vectors [Fig. 4, steps 2 and 2'] in the N^2 cells is therefore

$$T_i = 2(N^2TCI)TCI \quad (2)$$

For syndrome extraction [Fig. 4, steps 3 and 3'] performed through the TCO chain, the total time is

$$T_c = 2(N^2TCO)TCI \quad (3)$$

where the TCO 28 bit register must be extracted for each TCI vector injection. So the total crossbar test time is

$$T_{\text{crossbar}} = T_c + T_i + T_c, \quad (4)$$

$$T_{\text{crossbar}} = (N^2TCI)(CR + 2(TCI + TCO)) \quad (5)$$

B. Diagnosis time for the FPIN links

The total time for the crossbar configuration [Fig. 9 step (1)] is

$$T_c = (N^2CR) + (2N - 1)(2N)CR \quad (6)$$

where (N^2CR) is the initial time required once to configure all the cells not on the first diagonal in the bypass mode, while those on the first diagonal are configured as transmitter (Fig. 8 Conf 1). The $(2N)CR$ term corresponds to the size of the configuration register of the 2 diagonals to program, one in bypass mode and the other in transmitter mode. Finally the transmitter diagonal is shifted $(2N-1)$ times.

The total time for the walking test vector injection is:

$$T_i = 2N(2(N \times TCI + SHIFT)) \quad (7)$$

$$SHIFT = \frac{(TCI - 2)}{s} - 1 \quad (8)$$

The test vector is first loaded into the TCI register in $N \times TCI$ shift cycles. The loaded vector is then shifted $((TCI - 2)/s) - 1$ times, where s is the number of slices per TCI register. The TCI register can be divided in one, two (horizontal, vertical), or 4 (North, South, West, East) slices. The test has to be done for the 2 complemented test vectors and for the $2N$ configurations.

The total time for syndrome extraction is

$$T_c = 2N((N^2 - N)TCO)2(SHIFT + 1) \quad (9)$$

where $((N^2 - N)TCO)$ is the size of all the TCO registers to be extracted in the reticle, minus the TCO in the diagonal programmed as transmitter. The term $2(SHIFT + 1)$ is the number of tests deduced from (7).

The total time for FPIN link test is

$$T_{\text{link}} = T_c + T_i + T_c \quad (10)$$

Table 2 compares time complexities evaluated in test cycles to apply the diagnosis algorithms presented in [10] for $N=32$, which corresponds to the size used in the implemented VLAIC. It shows that our proposed diagnosis algorithms are 6.7, 129

and 113 time faster than those in [10] for the crossbar, the FPIN link and the total time diagnosis algorithms respectively.

C. Diagnosis complexity analysis

The time complexity of the diagnosis algorithms proposed in [10] was dominated by the FPIN link extraction of complexity $O(N^4)$. According to equation (5), our crossbar configuration time (T_c) and diagnosis time ($T_i + T_c$) algorithm are still $O(N^2)$. It is of interest that the reduction factor of 27.6 for crossbar syndrome extraction is due to the complexity coefficient that is reduced from 52 to 2. The time complexity of the proposed diagonal FPIN link diagnosis algorithm is still dominated by T_c (Eq. 10). The main contribution to the reduction factor is related to the FPIN links syndrome extraction time complexity from $O(N^4)$ to $O(N^3)$. This complexity cannot be further reduced and is $O(N^3)$.

Table. 2. Time complexity analysis of FPIN Diagnosis (cc= test clock cycles).

| | | [10] | Proposed | Reduction Factor [10]/proposed |
|---|-----------------------|-----------|-----------|-----------------------------------|
| Crossbar | T_c | 3.8 M cc | 3.8 M cc | 1 |
| | | $O(N^2)$ | $O(N^2)$ | |
| | T_i | 1.3 M cc | 1.3 M cc | 1 |
| | | $O(N^2)$ | $O(N^2)$ | |
| | T_c | 38.7 M cc | 1.4 M cc | 27.6 |
| FPIN Links | T_{crossbar} | 43.8 M cc | 6.5 M cc | 6.7 |
| | T_c | 1.32 M cc | 0.7 M cc | 1.9 |
| | | $O(N^2)$ | $O(N^2)$ | |
| | T_i | 3.4 M cc | 0.1 M cc | 34 |
| | | $O(N^2)$ | $O(N^2)$ | |
| | T_c | 5.6 G cc | 42.6 M cc | 131 |
| | | $O(N^4)$ | $O(N^3)$ | |
| | T_{link} | 5.6 G cc | 43.4 M cc | 129 |
| Total diagnosis time ($T_{\text{crossbar}} + T_{\text{link}}$) | | 5.64 G cc | 49.9 M cc | 113 |

IV. EXPERIMENTAL RESULTS

The proposed diagnosis algorithms were physically applied on the single available defect free reticle prototype fabricated in a 0.18 μm in CMOS technology (comprising 32×32 cells). It was also applied to a conventional packaged chip prototype comprising 8×7 cells. The JTAG bit stream was formatted and injected serially through an IGLOO AGL060 FPGA up to the device under test (Fig. 10 and 11). The FPGA is embedded in a power block module that provides mechanical support to reticles and supplies regulated power. The JTAG bit stream was generated from a Matlab® program, which also controls the VLAIC under test in our prototype environment. The resulting syndromes were saved in a file, and a post analysis was performed to derive final diagnostic.

Our VLAIC was designed such that a test can be applied at 50 MHz with the JTAG protocol. Due to limitations and

constraints of our custom experimental setup, test vector were injected through a low speed UART interface. Our diagnosis algorithm was then applied, the number of cycles measured and extrapolated for a 50 MHz test clock (TCK). Table 3 summarizes the results. It shows that only 33.6 ms are required to diagnose a reticle comprising 8×7 cells at 50 MHz. The diagnosis time for a reticle of 32×32 cells becomes one second for a test clock of 50 MHz rather than 2 minutes with the method proposed in [10].

The differences observed between calculated and measured values are due to the cycles needed to navigate through the different states of the JTAG FSM.

Table 3. Experimental diagnostic times in cycles and corresponding test times in second for the chip scale prototype comprising 8×7 cells.

| | Expected number of cycles | Measured number of cycles | Estimated test time @50 MHz |
|---------------|---------------------------|---------------------------|-----------------------------|
| Crossbar test | 36 692 | 39 125 | 0.0007s |
| Link test | 1 609 088 | 1 647 706 | 0.0329s |

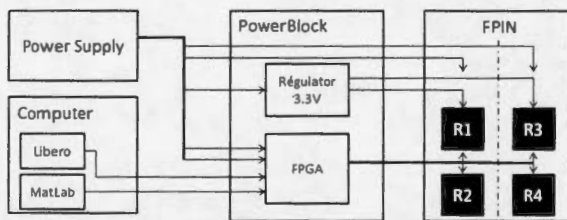


Fig. 10. VLAIC test bench.

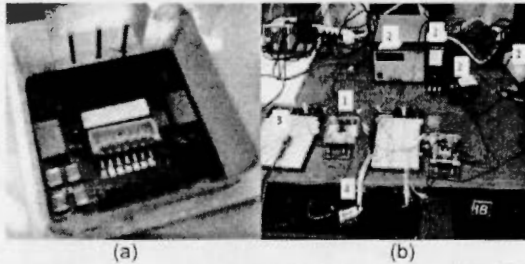


Fig. 11. VLAIC experimental environment, (a) test card FPIN placed under the Power Block. (b) Experimental setup: (1) test card, (2) four voltage supplies, (3) USB to serial UART converter, (4) FPGA programmer + JTAG cable.

V. CONCLUSION

In this paper, exhaustive and deterministic algorithms to diagnose defects in a crossbar based field programmable interconnect network implemented in a very large area integrated circuit was presented. As test time can induce significant costs, there is a need to minimize it. Also, good diagnosis algorithms producing accurate defect maps are needed to enable using partly defective configurable devices.

Specifically, we proposed a crossbar diagnosis algorithm that is 6.7 times faster than the one reported in [10]. We also proved that the generated test patterns induce syndromes that can be extracted and from which all single stuck-at and short

defects in a multiplexer-based crossbar I/Os can be diagnosed. The paper also proposed an efficient diagonal-based algorithm to diagnose single defects in links of the field programmable interconnect network, which reduces the cone of influence of individual tests and allows reducing the syndrome extraction time complexity from $O(N^4)$ to $O(N^3)$. That is significant as the syndrome extraction time dominates the diagnosis time. A diagnosis time analysis performed on a reticle image of 32×32 cells showed that our diagnosis algorithm is 113 times faster than that in [10]. Our diagnosis algorithm was also successfully applied on a packaged chip prototype comprising 8×7 cells.

ACKNOWLEDGMENT

The authors thank the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fonds de Recherche Nature et Technologies (FQRNT), MITACS and a Canada Research Chair to one of the authors for their financial support and CMC Microsystems for providing design tools and support. The work was also partly performed while one of the authors was a guest scientist at ParisTech Telecom (Comelec) and ENSEIRB (IMS-Bordeaux).

REFERENCES

- [1] R. Jain, A. Mukherjee, and K. Paul, "Defect-aware design paradigm for reconfigurable architectures," in *Emerging VLSI Technologies and Architectures*, 2006. IEEE Computer Society Annual Symposium on, 2006, p. 6.
- [2] Xilinx. EasyPath FPGAs. Available: <http://www.xilinx.com/products/silicon-devices/fpga/easypath-7.html>
- [3] A. Djupdal and P. C. Haddow, "Yield enhancing defect tolerance techniques for FPGAs," in *MAPLD International Conference*, 2006.
- [4] J. Brewer, "Promise and Pitfalls of WSI," in *Wafer Scale Integration*, E. Swartzlander, Jr., Ed., ed: Springer US, 1989, pp. 1-29.
- [5] D. Cohen, E. Koltin, M. I. Ayelet, and A. Shacham, "Stitching design rules for forming interconnect layers," U.S. Patent 6225013 B1, May 20, 1999.
- [6] F. C. Séverine Chéramy, Gilles Simon, Patrick Leduc, "The "active-interposer" concept for high-performance chip-to-chip connections," *ChipScale Review*, vol. 18, pp. 34-40, May-June, 2014.
- [7] S. Takaya, M. Nagata, A. Sakai, T. Kariya, S. Uchiyama, H. Kobayashi, et al., "A 100GB/s wide I/O with 4096b TSVs through an active silicon interposer with in-place waveform capturing," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2013 IEEE International, 2013, pp. 434-435.
- [8] R. Norman, O. Valorge, Y. Blaquiere, E. Lepercq, Y. Basile-Bellavance, Y. El-Alaoui, et al., "An active reconfigurable circuit board," in *Circuits and Systems and TAISA Conference*, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop on, 2008, pp. 351-354.
- [9] E. Lepercq, O. Valorge, Y. Basile-Bellavance, N. Laflamme-Mayer, Y. Blaquiere, and Y. Savaria, "An interconnection network for a novel reconfigurable circuit board," in *Microsystems and Nanoelectronics Research Conference*, 2009. MNRC 2009. 2nd, 2009, pp. 53-56.
- [10] Y. Basile-Bellavance, Y. Blaquiere, and Y. Savaria, "Faults diagnosis methodology for the WaferNet interconnection network," in *Circuits and Systems and TAISA Conference*, 2009. NEWCAS-TAISA '09. Joint IEEE North-East Workshop on, 2009, pp. 1-4.
- [11] E. Lepercq, Y. Blaquiere, R. Norman, and Y. Savaria, "Workflow for an electronic configurable prototyping system," in *Circuits and Systems*, 2009. ISCAS 2009. IEEE International Symposium on, 2009, pp. 2005-2008.
- [12] Y. Blaquiere, Y. Basile-Bellavance, S. Berrima, and Y. Savaria, "Design and validation of a novel reconfigurable and defect tolerant JTAG scan chain," in *Circuits and Systems (ISCAS)*, 2014 IEEE International Symposium on, 2014, pp. 2559-2562.